Principi matematici della robotica e dell'intelligenza artificiale incarnata

World Models, JEPA e le architetture dell'agire intelligente

PARTE I

FONDAMENTI MATEMATICI DELLA ROBOTICA

Fondamenti Matematici: Gruppi di Lie, Dinamica, Controllo e Strutture Neurali

Capitolo 1 — Strutture Algebriche per i Moti Rigidi: SO(3) e SE(3)

Il punto di partenza di ogni trattazione rigorosa della meccanica dei manipolatori è la caratterizzazione matematica dei moti rigidi nello spazio euclideo tridimensionale. Un moto rigido preserva le distanze mutue tra i punti del corpo e l'orientamento relativo dei sistemi di riferimento ad esso solidali; la struttura algebrica che incorpora simultaneamente entrambe queste proprietà è quella dei gruppi di Lie, ed è all'interno di questo linguaggio che le diverse componenti della cinematica, della dinamica e del controllo trovano la propria unità concettuale profonda.

La scelta di fondare l'intera costruzione sui gruppi di Lie non è meramente estetica. Quando si opera con manipolatori a sei o più gradi di libertà, con robot paralleli o con sistemi ridondanti, i formalismi elementari basati sugli angoli di Eulero rivelano patologie intrinseche — le singolarità della rappresentazione, concettualmente distinte dalle singolarità cinematiche — che rendono necessario lavorare direttamente sulle varietà di Lie. La derivazione di algoritmi efficienti per la dinamica ricorsiva, per la stima dello stato e per il controllo geometrico diventa trasparente e naturale soltanto nel linguaggio degli screw, dei torsori e delle corrispondenti algebre di Lie.

1.1 Il Gruppo delle Rotazioni SO(3)

Una trasformazione lineare $R : \mathbb{R}^3 \to \mathbb{R}^3$ è una rotazione propria se e soltanto se preserva il prodotto scalare euclideo e l'orientamento dello spazio. In termini matriciali queste due condizioni si scrivono come $R^{\top}R = I_3$ e $\det(R) = +1$. L'insieme di tutte le matrici $R \in \mathbb{R}^{3\times3}$ soddisfacenti tali proprietà, munito della moltiplicazione matriciale come operazione binaria, costituisce il **gruppo speciale ortogonale SO(3)**. La chiusura rispetto al prodotto, l'associatività, l'esistenza dell'elemento neutro I_3 e

dell'inverso — dato da $R^{-1} = R^{\top}$, proprietà computazionalmente preziosa — discendono direttamente dalla struttura algebrica.

SO(3) è al contempo una varietà differenziabile tridimensionale compatta e connessa, e le operazioni di prodotto e inversione sono applicazioni lisce su di essa: si è dunque in presenza di un **gruppo di Lie**. La topologia di SO(3) è non banale: il suo gruppo fondamentale $\pi_1(SO(3)) = \mathbb{Z}/2\mathbb{Z}$ attesta che la varietà non è semplicemente connessa. Il rivestimento universale è il gruppo SU(2) dei quaternioni unitari, con il quale SO(3) intrattiene una relazione di ricopertura 2:1 che ha conseguenze concrete nella programmazione dei manipolatori.

La mappa esponenziale e la formula di Rodrigues: derivazione completa

A ogni matrice antisimmetrica $\Omega \in \mathfrak{so}(3)$ è associata, tramite la mappa esponenziale, una rotazione in SO(3). L'algebra di Lie so(3) coincide con l'insieme delle matrici reali $A \in \mathbb{R}^{3\times3}$ tali che $A^{\top} = -A$, spazio vettoriale reale di dimensione tre. L'isomorfismo canonico tra $\mathbb{R}^3$ e so(3) è la *hat map* $[\cdot]^{\times} : \mathbb{R}^3 \to \mathfrak{so}(3)$, definita per $\omega = (\omega_1, \omega_2, \omega_3)^{\top}$ da

$$[\omega]^{\times} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{1.1}$$

L'azione di $[\omega]^{\times}$ su un vettore arbitrario $v \in \mathbb{R}^3$ riproduce il prodotto vettoriale: $[\omega]^{\times} v = \omega \times v$. La mappa $[\cdot]^{\times}$ è un isomorfismo di algebre di Lie tra $(\mathbb{R}^3, \times)$ e $(\mathfrak{so}(3), [\cdot,\cdot])$, dove il secondo membro denota il commutatore $[A, B] = AB - BA$. Esplicitamente, $[[\omega]^{\times}, [\eta]^{\times}] = [\omega \times \eta]^{\times}$.

Sia $\hat{k} \in S^2$ un versore ($\|\hat{k}\| = 1$) e $\theta \in \mathbb{R}$ un angolo. La matrice di rotazione di angolo θ attorno all'asse $\hat{k}$ è la soluzione al tempo $t = \theta$ dell'equazione differenziale ordinaria sul gruppo:

$$\dot{R}(t) = [\hat{k}]^{\times} R(t), \quad R(0) = I \tag{1.2}$$

La soluzione è data dall'esponenziale matriciale $R(t) = \exp(t\,[\hat{k}]_\times)$. Il calcolo esplicito di questo esponenziale si fonda sulle identità algebriche che le potenze di $[\hat{k}]_\times$ soddisfano. Poiché $[\hat{k}]_\times$ ha nucleo generato da $\hat{k}$ e immagine ortogonale ad esso, si ha:

$$[\hat{k}]_\times^2 = \hat{k}\,\hat{k}^\top - I, \quad [\hat{k}]_\times^3 = -[\hat{k}]_\times \tag{1.3}$$

Dimostrazione.

Per verificare la prima relazione, si applica $[\hat{k}]_\times$ due volte a un vettore arbitrario v: $[\hat{k}]_\times^2 v = \hat{k} \times (\hat{k} \times v) = \hat{k}(\hat{k}^\top v) - v(\hat{k}^\top \hat{k}) = (\hat{k}\hat{k}^\top - I)v$, dove si è usata l'identità di Lagrange per il triplo prodotto vettoriale. La seconda relazione segue immediatamente: $[\hat{k}]_\times^3 v = [\hat{k}]_\times([\hat{k}]_\times^2 v) = [\hat{k}]_\times(\hat{k}\hat{k}^\top - I)v = [\hat{k}]_\times(\hat{k}\hat{k}^\top v) - [\hat{k}]_\times v = 0 - [\hat{k}]_\times v$, poiché $[\hat{k}]_\times \hat{k} = \hat{k} \times \hat{k} = 0$.

□

Da (1.3) si ricava per induzione che $[\hat{k}]_\times^{2n+1} = (-1)^n [\hat{k}]_\times$ e $[\hat{k}]_\times^{2n} = (-1)^{n-1} [\hat{k}]_\times^2$ per $n \geq 1$. Separando i contributi a esponente pari e dispari nella serie di Taylor $\exp(\theta[\hat{k}]_\times) = I + \sum_{k=1}^{\infty} \theta^k [\hat{k}]_\times^k / k!$ e riconoscendo le serie di seno e coseno, si ottiene:

> **Teorema 1.1 (Formula di Rodrigues).** Per ogni versore $\hat{k} \in S^2$ e angolo $\theta \in \mathbb{R}$,
>
> $$\exp(\theta\,[\hat{k}]_\times) = I + \sin(\theta)\,[\hat{k}]_\times + (1 - \cos\theta)\,[\hat{k}]_\times^2 \tag{1.4}$$

Dimostrazione.

Raggruppando separatamente i termini di grado pari e dispari:

$$\exp(\theta\,[\hat{k}]_\times) = I + \sum_{n=0}^{\infty} (-1)^n \theta^{2n+1} [\hat{k}]_\times^{2n+1}/(2n+1)! + \sum_{n=1}^{\infty} (-1)^{n-1} \theta^{2n} [\hat{k}]_\times^{2n}/(2n)!$$

$$= I + [\hat{k}]_\times \sum_{n=0}^{\infty} (-1)^n \theta^{2n+1}/(2n+1)! + [\hat{k}]_\times^2 \sum_{n=1}^{\infty} (-1)^{n-1} \theta^{2n}/(2n)!$$

$$= I + \sin(\theta)\,[\hat{k}]_\times + (1 - \cos\theta)\,[\hat{k}]_\times^2$$

dove si sono riconosciute le serie di Taylor di $\sin\theta = \sum (-1)^n \theta^{2n+1}/(2n+1)!$ e $1 - \cos\theta = \sum (-1)^{n-1} \theta^{2n}/(2n)!$ con $n \geq 1$.

□

Quaternioni unitari e la copertura SU(2) → SO(3)

Un quaternione $q \in \mathbb{H}$ si scrive $q = q_0 + q_1 i + q_2 j + q_3 k$, dove q_0 è la parte scalare e $q_v = (q_1, q_2, q_3)^\top$ la parte vettoriale, con le relazioni $i^2 = j^2 = k^2 = ijk = -1$. Il prodotto di due quaternioni p e q è

$$p \otimes q = (p_0 q_0 - p_v \cdot q_v) + (p_0 q_v + q_0 p_v + p_v \times q_v) \quad (1.5)$$

Un quaternione unitario soddisfa $\|q\|^2 = q_0^2 + \|q_v\|^2 = 1$. L'insieme dei quaternioni unitari forma $S^3 \cong SU(2)$, e la mappa di copertura $\varrho : SU(2) \to SO(3)$ associa a $q = (\cos(\theta/2), \sin(\theta/2)\hat{k})$ la matrice

$$R(q) = (q_0^2 - \|q_v\|^2)\, I + 2\, q_v\, q_v^\top + 2 q_0\, [q_v]_\times \quad (1.6)$$

La stessa rotazione è codificata da q e $-q$, donde la struttura di ricopertura 2:1 (con nucleo $\{\pm I\} \cong \mathbb{Z}/2\mathbb{Z}$). L'interpolazione sferica lineare (SLERP) tra due orientamenti $q_1, q_2 \in S^3$ è

$$\mathrm{SLERP}(q_1, q_2; t) = q_1 \otimes (q_1^{-1} \otimes q_2)^t, \quad t \in [0,1] \quad (1.7)$$

e produce la geodetica (curva di minima lunghezza) su S^3 dotato della metrica bi-invariante, corrispondente alla traiettoria di minima lunghezza su SO(3). Questa proprietà variazionale è fondamentale nella pianificazione di moti rotativi ottimi.

1.2 Il Gruppo SE(3) e le Trasformazioni Omogenee

Un moto rigido in $\mathbb{R}^3$ comprende simultaneamente rotazione e traslazione. Il **gruppo speciale euclideo SE(3)** è il prodotto semidiretto $SO(3) \ltimes \mathbb{R}^3$: i suoi elementi sono coppie (R, t) con $R \in SO(3)$ e $t \in \mathbb{R}^3$, e la legge di composizione è $(R_1, t_1) \cdot (R_2, t_2) = (R_1 R_2, R_1 t_2 + t_1)$. La rappresentazione come matrice omogenea 4×4 è

$$T = \begin{bmatrix} R & t \\ 0^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \quad (1.8)$$

con inverso $T^{-1} = [[R^\top, -R^\top t]; [0^\top, 1]]$. La composizione di trasformazioni successive corrisponde al prodotto matriciale. L'algebra di Lie se(3) è costituita dai **torsori cinematici** (twists), matrici 4×4 della forma

$$[\xi] = \begin{bmatrix} [\omega]^\times & v \\ 0^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times4} \quad (1.9)$$

dove $\omega \in \mathbb{R}^3$ è la velocità angolare e $v \in \mathbb{R}^3$ è la velocità lineare del punto dell'organo terminale istantaneamente coincidente con l'origine. Il torsore può essere identificato con il vettore $\xi = (v^\top, \omega^\top)^\top \in \mathbb{R}^6$. La mappa esponenziale $\exp : se(3) \to SE(3)$ per un torsore $\hat{\xi}$ unitario e parametro θ produce il **moto elicoidale** (screw motion):

$$\exp(\theta[\hat{\xi}]) = \begin{bmatrix} \exp(\theta[\hat{\omega}]^\times) & A(\hat{\omega},\theta)\, v \\ 0^\top & 1 \end{bmatrix} \quad (1.10)$$

con $A(\hat{\omega}, \theta) = I\theta + (1 - \cos\theta)[\hat{\omega}]^\times + (\theta - \sin\theta)[\hat{\omega}]^2$ per $\|\hat{\omega}\| = 1$. Per $\omega = 0$ il moto degenera in una traslazione pura $\exp(\theta[\xi]) = [[I, v\theta]; [0, 1]]$; quando $v = h\omega$ (con h il *passo* dello screw), il moto è una rotazione pura attorno a un asse non passante per l'origine.

1.3 La Mappa Aggiunta e la Co-Aggiunta

La struttura di gruppo di Lie di SE(3) è arricchita da due azioni canoniche sulle algebre e le co-algebre di Lie, che costituiscono lo strumento fondamentale per trasformare torsori e forze generalizzate tra sistemi di riferimento diversi. Per $T = (R, t) \in SE(3)$, la **rappresentazione aggiunta** $\mathrm{Ad}_T : se(3) \to se(3)$ agisce sul torsore $\xi = (v, \omega)$ come

$$\mathrm{Ad}_T\, \xi = \begin{bmatrix} R & [t]^\times R \\ 0 & R \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} Rv + t \times R\omega \\ R\omega \end{bmatrix} \quad (1.11)$$

La mappa aggiunta trasforma il torsore dal sistema di riferimento del corpo al sistema di riferimento spaziale (o viceversa, a seconda della convenzione). La proprietà fondamentale è la **compatibilità con il**

prodotto di gruppo: $\mathrm{Ad}_{T_1T_2} = \mathrm{Ad}_{T_1}\,\mathrm{Ad}_{T_2}$. La derivata di Ad_T nell'identità restituisce la **rappresentazione aggiunta piccola** $\mathrm{ad}_\xi : \mathrm{se}(3) \to \mathrm{se}(3)$, definita dal commutatore:

$$\mathrm{ad}_\xi\,\eta = [\xi, \eta] = \xi\eta - \eta\xi \quad \in \mathrm{se}(3) \tag{1.12}$$

esplicitamente, per $\xi = (v_1, \omega_1)$ e $\eta = (v_2, \omega_2)$:

$$\mathrm{ad}_\xi\,\eta = (\omega_1 \times v_2 - \omega_2 \times v_1,\ \omega_1 \times \omega_2) \tag{1.13}$$

La **co-aggiunta** $\mathrm{Ad}^*_T : \mathrm{se}^*(3) \to \mathrm{se}^*(3)$ agisce sullo spazio duale $\mathrm{se}^*(3)$ — lo spazio dei bitorsori (wrenches) che raccoglie forze e momenti — trasponendo la mappa aggiunta: $\mathrm{Ad}^*_T = \mathrm{Ad}^{-\top}_T$. Questa dualità è alla base del principio di potenza virtuale e garantisce la covarianza delle equazioni di Lagrange rispetto a cambiamenti di sistema di riferimento.

Capitolo 2 — Cinematica dei Manipolatori e Convenzione di Denavit-Hartenberg

La cinematica dei manipolatori studia le relazioni geometriche tra lo spazio delle configurazioni e lo spazio operativo dell'organo terminale, prescindendo dalle forze che determinano il moto. Il formalismo DH, benché superato dal punto di vista teorico dalla formulazione a prodotto di esponenziali (trattata nel Capitolo 5), rimane lo standard industriale e merita una trattazione completa.

2.1 Lo Spazio delle Configurazioni come Varietà

Per un manipolatore seriale con n giunti, lo spazio delle configurazioni è la varietà $Q = Q_1 \times \cdots \times Q_n$, dove $Q_i \cong S^1$ per un giunto rotoidale e $Q_i \cong \mathbb{R}$ per uno prismatico. Nel caso esclusivamente rotoidale, $Q = (S^1)^n$ è il toro n-dimensionale, varietà compatta che riflette la periodicità degli angoli articolari. Lo spazio tangente $T_qQ \cong \mathbb{R}^n$ in ogni punto è lo spazio delle velocità articolari, e il fibrato tangente TQ è lo spazio degli stati cinematici $(q, \dot{q})$.

2.2 Parametri e Matrici di Denavit-Hartenberg

Per ogni giunto i si definisce il sistema di riferimento (x_i, y_i, z_i, o_i) secondo le convenzioni: z_i allineato all'asse del giunto i+1, x_i diretto lungo la normale comune agli assi z_{i-1} e z_i, o_i nell'intersezione di x_i con z_i. I quattro parametri DH sono: $\mathbf{a_i}$ (lunghezza del link), $\boldsymbol{\alpha_i}$ (torsione del link, angolo da z_{i-1} a z_i attorno a x_i), $\mathbf{d_i}$ (offset del giunto, distanza lungo z_{i-1}), $\boldsymbol{\theta_i}$ (angolo del giunto, da x_{i-1} a x_i attorno a z_{i-1}).

La trasformazione omogenea dal frame i–1 al frame i si ottiene componendo quattro trasformazioni elementari:

$$^{i-1}T_i = \mathrm{Rot}(z, \theta_i) \cdot \mathrm{Trans}(z, d_i) \cdot \mathrm{Trans}(x, a_i) \cdot \mathrm{Rot}(x, \alpha_i) \quad (2.1)$$

eseguendo esplicitamente le quattro moltiplicazioni:

$$ {}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2) $$

dove $c\theta_i = \cos\theta_i$ e $s\theta_i = \sin\theta_i$. La cinematica diretta del manipolatore si ottiene componendo in cascata *n* matrici DH: ${}^{0}T_n(q) = {}^{0}T_1(q_1) \cdot {}^{1}T_2(q_2) \cdot \ldots \cdot {}^{n-1}T_n(q_n)$.

2.3 Lo Jacobiano Geometrico: Struttura e Calcolo

Lo Jacobiano geometrico $J(q) \in \mathbb{R}^{6\times n}$ lega le velocità articolari alla velocità dell'organo terminale espressa nel frame di base. A differenza dello Jacobiano analitico $\partial f/\partial q$, il Jacobiano geometrico è definito direttamente in termini delle velocità angolari fisiche e non dipende dalla parametrizzazione dell'orientamento. La relazione fondamentale è

$$ \dot{x} = J(q)\,\dot{q}, \quad \dot{x} = (\dot{p}^{\top}, \omega^{\top})^{\top} \in \mathbb{R}^6 \quad (2.3) $$

Le colonne dello Jacobiano geometrico si determinano per contributo di ciascun giunto. Per un **giunto rotoidale** *i*:

$$ J_i(q) = \begin{bmatrix} z_{i-1} \times (p - p_{i-1}) \\ z_{i-1} \end{bmatrix} \quad (2.4) $$

Per un **giunto prismatico** *i*:

$$ J_i(q) = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \quad (2.5) $$

I vettori z_{i-1} e p_{i-1} si estraggono dalla matrice ${}^{0}T_{i-1}(q)$: la terza colonna della sottomatrice di rotazione fornisce z_{i-1}, mentre la colonna traslazione fornisce p_{i-1}. Il calcolo dello Jacobiano richiede le prime *n* trasformazioni parziali ed è eseguito ricorsivamente.

Singolarità cinematiche e indice di manipolabilità

Le singolarità dello Jacobiano si verificano nelle configurazioni per cui $\text{rank}(J(q)) < \min(6, n)$. Si classificano in **singolarità di confine** (robot completamente esteso o retratto) e **singolarità interne** (allineamento di assi articolari all'interno dello spazio di lavoro). L'indice di manipolabilità scalare di Yoshikawa misura la distanza dalla degenerazione:

$$w(q) = \sqrt{(\det(J(q)\, J^{\top}(q)))} \quad \geq \; 0 \tag{2.6}$$

con $w(q) = 0$ se e solo se q è una singolarità. In prossimità di una singolarità, la cinematica inversa diventa mal condizionata: velocità operative finite richiedono velocità articolari arbitrariamente elevate. Questo fenomeno giustifica l'impiego dello **pseudo-inverso smorzato** (damped least squares) $J^{\#} = J^{\top}(JJ^{\top} + \lambda^2 I)^{-1}$, con $\lambda > 0$ parametro di smorzamento che limita l'amplificazione delle velocità articolari a costo di un piccolo errore nell'inseguimento della velocità operativa.

2.4 Cinematica Inversa Numerica e Ridondanza

Il problema della cinematica inversa consiste nel determinare $q \in Q$ tale che $f(q) = T_d \in SE(3)$. La soluzione iterativa per il metodo dello pseudo-inverso esegue l'aggiornamento

$$q_{k+1} = q_k + J^{\dagger}(q_k)\, e_k \tag{2.7}$$

dove $J^{\dagger} = J^{\top}(JJ^{\top})^{-1}$ è lo pseudo-inverso destro di Moore-Penrose ed e_k è l'errore di posa in $\mathbb{R}^6$. Per robot ridondanti ($n > 6$), la soluzione generale è

$$\dot{q} = J^{\dagger}(q)\, \dot{x} + (I - J^{\dagger}J)\, z \tag{2.8}$$

Il secondo addendo è un moto articolare nel null-space di J, che non produce effetti sulla velocità dell'organo terminale. Il vettore $z \in \mathbb{R}^n$ può essere scelto per ottimizzare criteri secondari — questo meccanismo è trattato sistematicamente nel Capitolo 5.

2.5 Workspace, Singolarità e Analisi della Manipolabilità

Il workspace di un manipolatore è l'insieme dei punti raggiungibili dall'organo terminale nello spazio cartesiano. Si distinguono il workspace raggiungibile W_R = {p ∈ R^3 : ∃ q ∈ C, f_k(q) = p} e il workspace destro W_D ⊂ W_R, dove il robot può raggiungere il punto p con orientamento arbitrario. Per un manipolatore planare RR con lunghezze dei link l_1, l_2, W_R è l'anello {p : |l_1 - l_2| ≤ ‖p‖ ≤ l_1 + l_2}; la presenza o meno di W_D ≠ ∅ dipende dal rapporto l_1/l_2. Per i manipolatori industriali a 6 giunti, la struttura di W_R dipende dall'analisi delle sottomatrici dello Jacobiano e dalla presenza di singolarità.

Le singolarità cinematiche si verificano nelle configurazioni in cui lo Jacobiano J(q) perde rango. Formalmente, q è singolare se $\det(J(q)J(q)^T) = 0$, ovvero se σ_min(J(q)) = 0. Nelle singolarità di polso (wrist singularity), due assi di rotazione diventano allineati; nelle singolarità di spalla (shoulder singularity), il polso cade sull'asse del primo giunto. In prossimità di una singolarità, piccoli spostamenti nello spazio operativo richiedono velocità articolari illimitate: $\dot{p} = J^+(q)\dot{x}_e$, con $\|J^+(q)\| \to \infty$ quando σ_min → 0. Il metodo della singolarità robusta (Nakamura e Hanafusa 1986) introduce una regolarizzazione smorzata $J_\lambda = J^T(JJ^T + \lambda I)^{-1}$ che limita le velocità al costo di un errore di tracciamento controllato.

La manipolabilità $\omega(q) = \sqrt{(\det(J(q)J(q)^T))}$ misura quantitativamente la distanza dalla singolarità e la capacità del robot di muoversi con uguale facilità in tutte le direzioni. L'ellissoide di manipolabilità (Yoshikawa 1985) ha semi-assi σ_1 ≥ σ_2 ≥ ... ≥ σ_m (valori singolari di J) e assi nella direzione dei corrispondenti vettori singolari sinistri. Una forma alternativa è il condition number κ(J) = σ_max/σ_min, che misura il rapporto di anisotropia: κ = 1 (configurazione isotropa) indica la massima dexterity; κ → ∞ indica prossimità alla singolarità. L'ottimizzazione di criteri di manipolabilità tramite il null-space dello Jacobiano (sezione 2.4)

consente di massimizzare ω durante l'esecuzione di un task cartesiano, migliorando la robustezza del controllo.

2.6 Calibrazione Cinematica e Identificazione dei Parametri DH

I parametri DH nominali (d_i, a_i, α_i, θ_i) di un robot fisico differiscono dai valori di progetto a causa di tolleranze di lavorazione, deformazioni elastiche e wear degli attuatori. La calibrazione cinematica stima i parametri reali θ_DH minimizzando la discrepanza tra le posizioni misurate dell'organo terminale x_m e le posizioni predette dalla cinematica diretta f_k(q; θ_DH):

$$\theta^*_{DH} = \arg\min_\theta \sum_{k=1}^{N} \| x_m^{(k)} - f_k(q^{(k)}; \theta) \|^2 \quad (2.6)$$

Il problema è non lineare nei parametri θ, e viene tipicamente risolto mediante Gauss-Newton o Levenberg-Marquardt. La matrice Jacobiana del problema di calibrazione J_cal = ∂f_k/∂θ ha struttura sparsa (i parametri del giunto i influenzano solo i giunti i, i+1, ..., n) e il suo rango determina l'identificabilità del set di parametri. Il modello DH standard è singolare per giunti con assi paralleli (α_i = 0): in questo caso si usa il modello Hayati-Mirmirani (1985) che sostituisce il parametro d con un angolo di skew β. La calibrazione è fondamentale per la robotica industriale di precisione: un robot UR10 calibrato raggiunge precisioni di posizionamento di 0.1-0.3 mm, contro i 1-2 mm del modello nominale non calibrato.

Capitolo 3 — Dinamica Lagrangiana e Ricorsiva dei Manipolatori

3.1 Formulazione Lagrangiana e Matrice di Inerzia

Si scelgono le variabili articolari $q = (q_1, \ldots, q_n)^T$ come coordinate generalizzate. La funzione di Lagrange è $L(q, \dot{q}) = T(q, \dot{q}) - V(q)$, con T energia cinetica totale e V energia potenziale gravitazionale. Le equazioni di Eulero-Lagrange, per $i = 1, \ldots, n$, sono:

$$d/dt\,(\partial L/\partial \dot{q}_i) - \partial L/\partial q_i = \tau_i \quad (3.1)$$

L'energia cinetica del link i è $T_i = \tfrac{1}{2}\, m_i \,\|\dot{p}_{ci}\|^2 + \tfrac{1}{2}\, \omega_i^T\, I_{ci}\, \omega_i$, dove m_i è la massa, p_{ci} la posizione del baricentro e I_{ci} il tensore di inerzia rispetto al baricentro nel frame di base. Poiché $\dot{p}_{ci} = J_{vi}(q)\,\dot{q}$ e $\omega_i = J_{\omega i}(q)\,\dot{q}$, con J_{vi} e $J_{\omega i}$ gli Jacobiani del baricentro del link i, l'energia cinetica totale è $T(q, \dot{q}) = \tfrac{1}{2}\, \dot{q}^T M(q)\, \dot{q}$, con matrice di inerzia

$$M(q) = \Sigma_{i=1}^{n}\, \left(m_i\, J_{vi}^T(q)\, J_{vi}(q) + J_{\omega i}^T(q)\, R_i(q)\, \bar{I}_{ci}\, R_i^T(q)\, J_{\omega i}(q)\right) \in \mathbb{R}^{n \times n} \quad (3.2)$$

La matrice $M(q)$ è simmetrica e definita positiva (sotto ipotesi fisiche ragionevoli di inerzia non nulla in ogni direzione articolare). Svolgendo le derivate parziali di L nella (3.1) si ottiene la **equazione del manipolatore**:

$$M(q)\,\ddot{q} + C(q, \dot{q})\,\dot{q} + g(q) = \tau \quad (3.3)$$

Simboli di Christoffel e matrice di Coriolis

La matrice di Coriolis $C(q, \dot{q})$ si costruisce dai simboli di Christoffel del primo tipo associati alla metrica riemanniana indotta da $M(q)$ sullo spazio delle configurazioni:

$$c_{ijk}(q) = \tfrac{1}{2}\,(\partial m_{ij}/\partial q_k + \partial m_{ik}/\partial q_j - \partial m_{jk}/\partial q_i) \quad (3.4)$$

donde $C_{ij}(q, \dot{q}) = \Sigma_{k=1}^{n}\, c_{ijk}(q)\, \dot{q}_k$. La scelta basata sui simboli di Christoffel è l'unica che garantisce la fondamentale proprietà strutturale:

> **Proposizione 3.1 (Simmetria emisimmetrica).** Con la definizione (3.4), la matrice $(\dot{M}(q, \dot{q}) - 2C(q, \dot{q}))$ è **antisimmetrica**, ovvero

$$\dot{q}^{\top} (\dot{M} - 2C)\, \dot{q} = 0 \quad \forall\, \dot{q} \in \mathbb{R}^n \tag{3.5}$$

Dimostrazione.

Si calcola l'elemento (i, j) di $\dot{M} - 2C$. La derivata temporale di $m_{ij}(q)$ è $\dot{m}_{ij} = \Sigma_k (\partial m_{ij}/\partial q_k)\, \dot{q}_k$. Quindi $(\dot{M} - 2C)_{ij} = \Sigma_k (\partial m_{ij}/\partial q_k - \partial m_{ik}/\partial q_k - \partial m_{jk}/\partial q_i)\, \dot{q}_k$. L'antisimmetria in i, j di questo tensore segue dall'antisimmetria del fattore $(\partial m_{ij}/\partial q_k - \partial m_{ik}/\partial q_k - \partial m_{jk}/\partial q_i)$ rispetto allo scambio $i \leftrightarrow j$, che si verifica per ispezione diretta usando la simmetria $m_{ij} = m_{ji}$. Quindi $\dot{q}^{\top}(\dot{M} - 2C)\dot{q} = \Sigma_{ij} (\dot{M} - 2C)_{ij}\, \dot{q}_i \dot{q}_j = 0$ per antisimmetria del fattore tensoriale.

□

Questa proprietà implica la **passività** del sistema robotico: scegliendo $V_s(q, \dot{q}) = \tfrac{1}{2}\, \dot{q}^{\top} M(q) \dot{q} + V(q)$ come funzione di stoccaggio, si verifica $\dot{V}_s = \tau^{\top} \dot{q}$, confermando che la potenza iniettata al sistema corrisponde all'incremento dell'energia totale, senza generazione interna.

3.2 L'Algoritmo Newton-Euler Ricorsivo

L'algoritmo ricorsivo di Newton-Euler (RNEA) calcola le coppie generalizzate in risposta a un moto articolare assegnato con complessità O(n), grazie a due sweep ricorsivi. Nel **primo sweep** (base → organo terminale) si propagano velocità e accelerazioni:

$$^{i}\omega_i = {}^{i}R_{i-1}\, {}^{i-1}\omega_{i-1} + \dot{\theta}_i\, {}^{i}\hat{z}_i \quad \text{(giunto rotoidale)} \tag{3.6}$$

$$^{i}\dot{\omega}_i = {}^{i}R_{i-1}\, {}^{i-1}\dot{\omega}_{i-1} + \ddot{\theta}_i\, {}^{i}\hat{z}_i + \dot{\theta}_i\, {}^{i}\omega_i \times {}^{i}\hat{z}_i \tag{3.7}$$

Nel **secondo sweep** (organo terminale → base) si propagano forze e momenti, ottenendo alla fine la coppia del giunto i come $\tau_i = {}^{i}n_i^{\top}\, {}^{i}\hat{z}_i$ (rotoidale) o $\tau_i = {}^{i}f_i^{\top}\, {}^{i}\hat{z}_i$ (prismatico). Il numero di operazioni è O(n) poiché ogni sweep richiede un numero costante di operazioni per link, e

l'algoritmo risulta superiore di svariati ordini di grandezza rispetto al calcolo diretto di M(q), C(q,q̇) e g(q) per $n > 6$.

3.3 Identificazione dei Parametri Inerziali

La dinamica del manipolatore M(q)q̈ + C(q,q̇)q̇ + g(q) = τ è lineare nel vettore dei parametri inerziali φ ∈ R^{10n} (per un robot a n giunti, ciascun link ha 10 parametri: massa, tre momenti statici, sei componenti del tensore di inerzia). Questa proprietà di linearità nei parametri — la stessa sfruttata dal controllo adattativo di Slotine-Li (sezione 4.5) — permette di scrivere:

$$\tau = W(q, \dot{q}, \ddot{q})\, \varphi \quad (3.5)$$

dove W(q, q̇, q̈) ∈ R^{n×10n} è la matrice di regressione, costruita esplicitamente dalle espressioni simboliche della dinamica. L'identificazione del vettore φ dai dati sperimentali τ_m (coppia misurata), q_m, q̇_m, q̈_m si riduce a un problema di minimi quadrati lineari:

$$\varphi^* = (W^T W)^{-1} W^T \tau_m \quad (3.6)$$

La condizione di osservabilità richiede che W^TW sia non singolare, il che impone vincoli sulle traiettorie di eccitazione usate per la raccolta dati. Le traiettorie ottimali di eccitazione (Swevers et al. 1997) massimizzano il minimo valore singolare di W rispetto alla forma della traiettoria (tipicamente somme di Fourier troncate), garantendo che tutti i parametri siano ben identificabili. Nella pratica industriale, la qualità dell'identificazione dipende criticamente dalla frequenza di campionamento, dal filtraggio delle misure di coppia (che presentano rumore ad alta frequenza) e dalla stima delle accelerazioni articolari q̈ (non misurate direttamente, ma calcolate per differenziazione numerica filtrata). Un errore sistematico in q̈ produce un bias nell'identificazione di M(q) che si propaga sull'intero vettore φ.

Il set di parametri fisicamente ammissibili (physically consistent) richiede che il tensore di inerzia di ciascun link sia semidefinito positivo e che la massa sia positiva. La violazione di questi vincoli produce parametri identificati che, pur minimizzando l'errore di fitting, non corrispondono a corpi fisici reali e possono portare a instabilità nel controllo. I metodi di identificazione vincolata (Wensing et al. 2018) formulano il problema come un SDP (semidefinite program) che garantisce la consistenza fisica dei parametri stimati.

3.4 Struttura Porto-Hamiltoniana e Passività

L'equazione dinamica del manipolatore può essere riscritta nella forma porto-hamiltoniana che mette in luce la struttura energetica del sistema. Definendo lo stato $(q, p) = (q, M(q)\dot{p}) \in T^*Q$ (fibrato cotangente della varietà delle configurazioni) e l'Hamiltoniano $H(q,p) = \frac{1}{2}p^T M^{-1}(q)p + V(q)$ (somma di energia cinetica e potenziale), la dinamica si scrive:

$$d/dt\,[q; p] = [0, I; -I, -D]\,\nabla H(q,p) + [0; I]\,\tau \quad (3.7)$$

dove $D \geq 0$ è la matrice di smorzamento (attrito viscoso). La proprietà fondamentale di questa struttura è la passività: il sistema è passivo con funzione di storage $H(q,p) \geq 0$, ovvero la potenza estratta dal sistema non supera mai l'energia immagazzinata: $dH/dt \leq \tau^T\dot{p}$. Questa proprietà, dimostrata dalla diseguaglianza $dH/dt = \tau^T\dot{p} - \dot{p}^T D\dot{p} \leq \tau^T\dot{p}$, garantisce la stabilità dell'interazione con ambienti passivi: se il robot interagisce con un ambiente passivo (energia estratta dall'ambiente ≥ 0), il sistema complessivo rimane stabile indipendentemente dalla rigidezza dell'ambiente. Questa è la base teorica del controllo di impedenza (Capitolo 28) e del controllo adattativo (sezione 4.5): entrambi sfruttano la struttura porto-hamiltoniana per garantire proprietà di stabilità robuste.

Capitolo 4 — Fondamenti di Controllo dei Manipolatori

4.1 Controllo PD con Compensazione Gravitazionale

Il controllore più semplice che garantisce convergenza alla configurazione desiderata q_d è il controllo PD con compensazione del termine gravitazionale: $\tau = K_p e + K_d \dot{e} + g(q)$, con $e = q_d - q$. La dimostrazione di stabilità asintotica si conduce con la funzione di Lyapunov

$$V(e, \dot{e}) = \tfrac{1}{2}\, \dot{e}^T M(q)\, \dot{e} + \tfrac{1}{2}\, e^T K_p\, e \quad (4.1)$$

definita positiva per $M(q) > 0$ e $K_p > 0$. La sua derivata lungo le traiettorie del sistema in anello chiuso è $\dot{V} = -\dot{e}^T K_d \dot{e} \leq 0$, dove si è sfruttata la proprietà antisimmetrica di $\dot{M} - 2C$. Poiché $\dot{V} = 0 \Rightarrow \dot{e} = 0$, per il principio di invarianza di LaSalle si conclude $q \to q_d$ e $\dot{q} \to 0$ asintoticamente.

4.2 Controllo a Coppia Calcolata

La **linearizzazione ingresso-uscita** (o controllo a coppia calcolata) sfrutta la conoscenza del modello per trasformare il sistema nonlineare in integratori doppi disaccoppiati. La legge di controllo è

$$\tau = M(q)\, a + C(q, \dot{q})\, \dot{q} + g(q) \quad (4.2)$$

con segnale ausiliare $a = \ddot{q}_d + K_d \dot{e} + K_p e$. Sostituendo in (3.3), i termini nonlineari si cancellano esattamente e si ottiene la dinamica dell'errore:

$$\ddot{e} + K_d\, \dot{e} + K_p\, e = 0 \quad (4.3)$$

Con $K_p = \omega_n^2 I$ e $K_d = 2\zeta\omega_n I$, la convergenza è esponenziale con tasso $\zeta\omega_n$ e frequenza naturale ω_n liberamente assegnabili. Il limite fondamentale del metodo è la sensitività agli errori di modello: se la dinamica reale differisce dal modello utilizzato nel calcolo di τ, la cancellazione dei

termini nonlineari è solo approssimata. I controlli adattativi e robusti estendono questo approccio stimando on-line i parametri incerti, sfruttando la proprietà di **linearità nei parametri** del modello: $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = Y(q, \dot{q}, \ddot{q})\,\theta$, con $Y \in \mathbb{R}^{6xp}$ matrice regressore e $\theta \in \mathbb{R}^{p}$ vettore dei parametri dinamici (masse, momenti, distanze baricentri).

4.3 Il Regolatore LQR e l'Equazione di Riccati

Linearizzando l'equazione del manipolatore attorno a una configurazione di equilibrio $(q_0, 0)$ e ponendo $x = (e^{\top}, \dot{e}^{\top})^{\top} \in \mathbb{R}^{2n}$, il sistema ha la forma $\dot{x} = Ax + Bu$ con

$$A = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ M^{-1}(q_0) \end{bmatrix} \tag{4.4}$$

Il problema LQR minimizza il funzionale di costo quadratico $J = \int_0^{\infty} (x^{\top}Qx + u^{\top}Ru)\,dt$, con $Q = Q^{\top} \geqslant 0$ e $R = R^{\top} > 0$. La soluzione ottima $u^* = -Kx$ ha guadagno

$$K = R^{-1} B^{\top} P \tag{4.5}$$

dove $P = P^{\top} > 0$ è l'unica soluzione dell'equazione algebrica di Riccati (ARE):

$$PA + A^{\top}P - PBR^{-1}B^{\top}P + Q = 0 \tag{4.6}$$

L'esistenza e l'unicità di $P > 0$ sono garantite dalla stabilizzabilità di (A, B) e dalla rilevabilità di $(Q^{1/2}, A)$. Il controllore LQR possiede intrinseche proprietà di robustezza per il sistema lineare: margini di guadagno di almeno 6 dB e margini di fase di almeno 60°, risultati dovuti alla teoria della return difference matrix e alla disuguaglianza di Kalman.

4.4 Il Controllo Predittivo su Modello (MPC)

Il Model Predictive Control (MPC) estende l'approccio LQR a sistemi con vincoli sugli stati e sugli ingressi, risolvendo on-line un problema di ottimizzazione su un orizzonte di previsione finito *N*. Ad ogni istante *k*, si minimizza

$$J = \Sigma_{j=0}^{N-1} \; (x_{k+j}^{\top} Q \, x_{k+j} + u_{k+j}^{\top} R \, u_{k+j}) + x_{k+j}^{\top} P_{\infty} \, x_{k+j} \quad (4.7)$$

soggetto a $x_{k+j+1} = Ax_{k+j} + Bu_{k+j}$, $x \in \mathcal{X}$ e $u \in \mathcal{U}$. Il termine terminale P_{∞} (soluzione dell'ARE) garantisce la stabilità del sistema in anello chiuso con orizzonte finito. Solo il primo elemento della sequenza ottima u^*_0 viene applicato, e il problema viene risolto nuovamente al passo successivo (receding horizon). Per manipolatori robotici, l'MPC consente di gestire sistematicamente i limiti di coppia, di velocità articolare e di spazio di lavoro, a costo di una risoluzione iterativa di un problema di programmazione quadratica (QP) ad ogni ciclo di controllo.

4.5 Controllo Adattativo di Slotine-Li

Il controllo a coppia calcolata (§4.2) garantisce la cancellazione esatta dei termini nonlineari soltanto quando i parametri dinamici del manipolatore — masse, posizioni dei baricentri, tensori di inerzia — sono noti con precisione. In pratica questa ipotesi non è mai pienamente verificata: i parametri reali variano per usura, carico utile variabile, e deriva termica degli attuatori. Il controllo adattativo di Slotine e Li (1987) risolve questo problema stimando i parametri sconosciuti in tempo reale senza misurare accelerazioni articolari, sfruttando la proprietà di linearità nei parametri del modello dinamico.

Linearità nei parametri e il regressore

Come si è dimostrato nel §4.2, l'equazione del manipolatore può essere scritta nella forma regressore–parametri. Per un vettore di accelerazioni di riferimento $\ddot{q}_r \in \mathbb{R}^n$ e velocità di riferimento $\dot{q}_r \in \mathbb{R}^n$ (definiti di

seguito), esiste una matrice regressore $Y_r : Q \times TQ \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n \times p}$ tale che:

$$M(q)\ddot{q}_r + C(q, \dot{q})\dot{q}_r + g(q) = Y_r(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\,\theta \quad (4.8)$$

Il vettore $\theta \in \mathbb{R}^p$ raccoglie i parametri dinamici del robot (masse, momenti del primo e secondo ordine rispetto all'origine del frame di base di ciascun link). La matrice Y_r è calcolabile analiticamente a partire dalla cinematica e non dipende dai parametri incogniti. La chiave della formula (4.8) è che il lato sinistro è lineare nei parametri θ — non nei valori q, $\dot{q}$, $\dot{q}_r$, $\ddot{q}_r$, che possono apparirvi in modo altamente nonlineare.

Variabile di scorrimento e legge di controllo

Sia $e = q_d - q \in \mathbb{R}^n$ l'errore di inseguimento e si definiscano le traiettorie di riferimento modificate:

$$\dot{q}_r = \dot{q}_d + \Lambda e, \quad \ddot{q}_r = \ddot{q}_d + \Lambda \dot{e} \quad (4.9)$$

con $\Lambda = \Lambda^T > 0$ matrice di guadagno. La variabile di scorrimento $s \in \mathbb{R}^n$ è definita come:

$$s = \dot{e} + \Lambda e = \dot{q}_r - \dot{q} \quad (4.10)$$

Il vettore s misura la distanza dalla varietà attrattiva $\{(q, \dot{q}) : \dot{e} + \Lambda e = 0\}$; quando $s = 0$ l'errore converge esponenzialmente a zero con tasso determinato dagli autovalori di Λ. La legge di controllo adattativo è:

$$\tau = Y_r(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\,\hat{\theta} - K_d\, s \quad (4.11)$$

dove $\hat{\theta} \in \mathbb{R}^p$ è la stima corrente dei parametri e $K_d = K_d^T > 0$ è il guadagno dissipativo. La legge di aggiornamento adattativo, derivata dall'annullamento del gradiente della funzione di Lyapunov, è:

$$\dot{\hat{\theta}} = -\Gamma\, Y_r^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\, s \quad (4.12)$$

con $\Gamma = \Gamma^T > 0$ matrice di guadagno adattativo. La notazione $\tilde{\theta} = \hat{\theta} - \theta$ indica l'errore di stima.

Teorema di stabilità globale

> **Teorema 4.3 (Slotine–Li, 1987).** Per il manipolatore con dinamica $M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$ e la coppia di leggi (4.11)–(4.12), la funzione di Lyapunov candidata
>
> $$V(s,\tilde{\theta}) = \tfrac{1}{2}\, s^{T} M(q)\, s + \tfrac{1}{2}\,\tilde{\theta}^{T}\Gamma^{-1}\tilde{\theta} \geq 0 \quad (4.13)$$
>
> è non crescente lungo le traiettorie del sistema in anello chiuso, e la soluzione $(s,\tilde{\theta}) = (0,0)$ è stabile in senso di Lyapunov. Inoltre $s(t) \to 0$ e quindi $e(t) \to 0$, $\dot{e}(t) \to 0$ asintoticamente, mentre la stima $\hat{\theta}(t)$ rimane limitata per ogni $t \geq 0$.

Dimostrazione.

Si calcola la derivata temporale di V. Il primo termine richiede la derivata di $\tfrac{1}{2}\, s^{T} M(q)\, s$. Usando la dinamica del manipolatore e la legge di controllo (4.11):

$$M(q)\dot{s} = M(q)(\ddot{q}_r - \ddot{q}) = M(q)\ddot{q}_r - (\tau - C(q,\dot{q})\dot{q} - g(q))$$

$$= M(q)\ddot{q}_r + C(q,\dot{q})\dot{q} + g(q) - Y_r\,\hat{\theta} + K_d\, s$$

$$= Y_r\,\theta - Y_r\,\hat{\theta} + K_d\, s = -Y_r\,\tilde{\theta} + K_d\, s$$

dove si è usata (4.8) e la definizione $\tilde{\theta} = \hat{\theta} - \theta$. Ora si calcola $\dot{V}$:

$$\dot{V} = s^{T} M(q)\,\dot{s} + \tfrac{1}{2}\, s^{T}\dot{M}(q)\, s + \tilde{\theta}^{T}\Gamma^{-1}\dot{\hat{\theta}}$$

$$= s^{T}(-Y_r\,\tilde{\theta} + K_d\, s) + \tfrac{1}{2}\, s^{T}\dot{M}\, s + \tilde{\theta}^{T}\Gamma^{-1}(-\Gamma\, Y_r^{T}\, s)$$

$$= -s^{T} Y_r\,\tilde{\theta} + \tfrac{1}{2}\, s^{T}\dot{M}\, s - s^{T} K_d\, s - \tilde{\theta}^{T} Y_r^{T}\, s$$

I termini $-s^{T} Y_r\,\tilde{\theta}$ e $-\tilde{\theta}^{T} Y_r^{T}\, s$ si cancellano perché sono l'uno il trasposto dell'altro (entrambi scalari uguali). Applicando la proprietà antisimmetrica di $\dot{M} - 2C$ (Proposizione 3.1) si ha $s^{T}\dot{M}\, s = s^{T}(\dot{M} - 2C)s + 2s^{T} C\, s = 0 + 2s^{T} C\, s$... No, più precisamente: $s^{T}\dot{M}\, s = 2\, s^{T} C\, s$ poiché $s^{T}(\dot{M} - 2C)s = 0$ implica $s^{T}\dot{M}\, s = 2\, s^{T} Cs$. Il termine $s^{T} Cs$ si cancella con il termine

proveniente dall'espansione di $M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r$ rispetto a $\dot{q}$ e $\dot{q}_r$. Raccogliendo:

$$\dot{V} = -s^T K_d s \leq 0 \quad (4.14)$$

La funzione V è pertanto non crescente. Poiché V è limitata inferiormente da zero e non crescente, V(t) converge a un valore finito $V_\infty \geq 0$, il che implica che $s(t) \in L_2 \cap L_\infty$ e $\hat{\theta}(t) \in L_\infty$. Dal lemma di Barbalat, applicabile poiché $\dot{V}$ è limitata (si verifica che $\dot{s}$ e $\dot{\hat{\theta}}$ sono limitati dall'equazione del moto e dalla legge (4.12) con $s \in L_\infty$), si conclude $s(t) \to 0$. La convergenza di $s \to 0$ implica $e(t) \to 0$ ed $\dot{e}(t) \to 0$ per la definizione (4.10) e la stabilità del filtro $\dot{e} + \Lambda e = s$ con $s \to 0$.

□

Si osservi che il teorema non garantisce la convergenza della stima $\hat{\theta}(t)$ al valore vero θ: la stima rimane limitata ma può attestarsi su un qualsiasi valore compatibile con la condizione $s = 0$. La convergenza parametrica si ottiene soltanto sotto condizioni di eccitazione persistente (Persistency of Excitation, PE): il regressore Y_r deve eccitare sufficientemente il sistema, condizione verificabile a priori soltanto per traiettorie di riferimento sufficientemente ricche.

Condizione di Eccitazione Persistente: definizione formale e conseguenze

La seguente definizione precisa il requisito intuitivo di "ricchezza" della traiettoria di riferimento, traducendolo in una condizione algebrica sulla matrice di informazione del regressore.

> **Definizione 4.2 (Eccitazione Persistente).** Il regressore $Y_r : [0, \infty) \to \mathbb{R}^{n \times p}$ è persistentemente eccitato (PE) se esistono costanti $T_0 > 0$, $\alpha_1 > 0$ e $\alpha_2 > 0$ tali che per ogni $t \geq 0$:
>
> $$\alpha_1 I \leq \int_t^{t+T_0} Y_r^T(\tau)\, Y_r(\tau)\, d\tau \leq \alpha_2 I \quad (4.15)$$

La disuguaglianza inferiore in (4.15) è la condizione sostanziale: essa richiede che la matrice di informazione $\int Y\_r^T Y\_r \, d\tau$ abbia rango pieno p su ogni finestra temporale di lunghezza T_0, garantendo che il sistema sia sufficientemente stimolato in tutte le direzioni dello spazio dei parametri. La disuguaglianza superiore assicura semplicemente la limitatezza del regressore. Narendra e Annaswamy (1987) hanno dimostrato che la condizione PE (4.15) è sufficiente per la convergenza parametrica esponenziale: se Y_r è PE allora $\| \hat{\theta}(t) - \theta \| \leq c \, e^{-\lambda t} \| \hat{\theta}(0) - \theta \|$ per costanti $c, \lambda > 0$ che dipendono da α_1, T_0 e dalla matrice di guadagno Γ. In pratica, traiettorie sinusoidali multi-frequenza del tipo $q\_d(t) = \Sigma_i \, A_i \sin(\omega_i t + \varphi_i)$ con almeno $\lceil p/2n \rceil$ frequenze distinte soddisfano genericamente la PE, mentre traiettorie punto-a-punto su un singolo asse non la soddisfano perché eccitano solo una direzione del regressore.

4.6 Architetture di Controllo Gerarchico per Robot Mobili e Umanoidi

I robot mobili e i robot umanoidi richiedono architetture di controllo a più livelli che gestiscano simultaneamente la pianificazione ad alto livello (task planning), la generazione di traiettorie (motion planning) e il controllo a bassa latenza (feedback control). L'architettura gerarchica standard a tre livelli, adottata da sistemi come Atlas (Boston Dynamics), HRP-4 (AIST) e Spot, separa il problema in tre sottoproblemi con frequenze di aggiornamento diverse: il livello di task planning (1-10 Hz) genera obiettivi ad alto livello (es. "fai presa sull'oggetto A"); il livello di motion planning (10-100 Hz) calcola traiettorie nello spazio delle configurazioni; il livello di feedback control (1-5 kHz) realizza la traiettoria con i controllori derivati nei paragrafi precedenti.

Per i robot umanoidi il problema centrale del controllo a bassa latenza è il Whole-Body Control (WBC): determinare le coppie articolari $\tau \in R^n$ che soddisfano simultaneamente un insieme di task prioritizzati (mantenere il

bilanciamento, seguire una traiettoria dell'end-effector, evitare i limiti articolari) rispettando i vincoli di contatto con il suolo. Il WBC formula il problema come un QP (quadratic program) a priorità gerarchica (Sentis e Khatib 2005):

$$\min_{\tau, f_c} \|J_k \dot{p} - \dot{x}_{k,des}\|^2_{W_k} \quad \text{s.t. } M\dot{p} + C\dot{p} + g = \tau + J_c^T f_c,\ f_c \in FC \quad (4.7)$$

dove J_k è lo Jacobiano del task k-esimo, f_c ∈ R^{3m} sono le forze di contatto (m = numero di piedi a contatto), J_c è lo Jacobiano di contatto, e FC è il vincolo del cono di attrito (sezione 28.1 del Capitolo 28). La matrice di peso W_k regola la priorità: il bilanciamento (k=1) ha W_1 ≫ W_2 (tracking end-effector) ≫ W_3 (ottimizzazione postural). In pratica, i sistemi moderni come OSC (Operational Space Control) e TSID (Task Space Inverse Dynamics, Del Prete et al. 2016) risolvono questo QP a 1 kHz su hardware embedded, raggiungendo prestazioni comparabili a controllori analitici ma con maggiore flessibilità nella gestione delle priorità.

Per i robot mobili con base differenziale o omnidirezionale, il controllo gerarchico separa la pianificazione del percorso globale (Capitolo 27) dal controllo locale di traiettoria. Il controllore di traiettoria tipicamente implementa un inseguimento di curvatura basato sul modello cinematico del veicolo. Per un robot differenziale con larghezza del passo 2d, velocità lineari v e angolari ω, il modello cinematico è:

$$[\dot{x};\ \dot{y};\ \dot{\theta}] = [\cos\theta,\ 0;\ \sin\theta,\ 0;\ 0,\ 1]\ [v;\ \omega],\quad v = (v_R + v_L)/2,\ \omega = (v_R - v_L)/(2d) \quad (4.8)$$

Il controllo di traiettoria di Lyapunov (Samson 1995) costruisce una legge di retroazione che porta l'errore di posizionamento (e_x, e_y, e_θ) a zero per traiettorie di riferimento non stazionarie, garantendo la stabilità globale asintotica via funzione di Lyapunov V = e_x^2 + e_y^2 + (1 - cos e_θ)/k_θ per k_θ > 0. I robot umanoidi bipedalici richiedono un livello aggiuntivo per la gestione dell'equilibrio dinamico: il modello lineare del pendolo invertito (LIPM) approssima la dinamica del centro di massa

(CoM) come un sistema lineare governato dalla posizione del Centro di Pressione (CoP), consentendo di pianificare passi stabili tramite MPC (sezione 4.4) a 100 Hz.

4.6.1 Connessioni con i World Model e l'Architettura JEPA

L'architettura gerarchica descritta in questo paragrafo trova la sua controparte nelle architetture di AI incarnata analizzate nelle Parti III-IV. Il world model latente (Parte III) funge da livello di pianificazione: predice le conseguenze delle azioni ad alto livello e guida la selezione degli obiettivi. Il controllore WBC o il controllore di traiettoria (questo capitolo) funge da livello di esecuzione: realizza fisicamente le azioni selezionate dal world model. La separazione tra pianificazione latente e controllo fisico corrisponde alla visione di LeCun del framework JEPA gerarchico (Capitolo 24): i livelli superiori della gerarchia JEPA pianificano in spazi latenti astratti, mentre i livelli inferiori (i controllori di questo capitolo) operano nello spazio fisico delle forze e delle posizioni articolari. I sistemi VLA di quarta generazione come GR00T N1 (Capitolo 29) implementano concretamente questa separazione: il modello foundation (VLM backbone) svolge il ruolo del livello di task planning, mentre le policy diffusive o autoregressivo svolgono il ruolo del motion planning, lasciando al WBC fisico la chiusura del ciclo a bassa latenza.

Capitolo 5 — Il Metodo del Prodotto di Esponenziali, Jacobiano Spaziale e Analisi SVD della Manipolabilità

La formulazione del Prodotto di Esponenziali (PoE), sviluppata in modo sistematico da Murray, Li e Sastry nell'opera fondamentale del 1994, offre un'alternativa intrinsecamente geometrica alla convenzione DH. I suoi vantaggi risiedono nell'assenza di singolarità della parametrizzazione, nella dipendenza esclusiva dalla geometria dello screw di ciascun giunto, e nella derivazione coordinata dello Jacobiano spaziale tramite la mappa aggiunta del gruppo SE(3) — derivazione che illumina la struttura algebrica sottostante a ogni manipolatore seriale.

5.1 La Configurazione di Riferimento e gli Screw Assi

Si sceglie una **configurazione di riferimento (home)** del robot, tipicamente corrispondente a $q = 0$, in cui la posa dell'organo terminale nel frame di base è $T_{home} \in SE(3)$. In questa configurazione, a ogni giunto i si associa il corrispondente **screw axis** $\xi_i = (v_i, \omega_i) \in \mathbb{R}^6$, dove:

Per un giunto rotoidale con asse che passa per il punto $q_i \in \mathbb{R}^3$ nel frame di base e ha direzione $\hat{\omega}_i \in S^2$:

$$\xi_i = (-\hat{\omega}_i \times q_i,\ \hat{\omega}_i) \quad \in \mathbb{R}^6 (5.1)$$

Per un giunto prismatico con direzione di traslazione $\hat{d}_i \in S^2$:

$$\xi_i = (\hat{d}_i,\ 0) \quad \in \mathbb{R}^6 (5.2)$$

Questi screw axes sono definiti in via permanente nella configurazione di riferimento e non dipendono dalla configurazione corrente del robot: è questa invarianza che rende il formalismo PoE superiore alla convenzione DH dal punto di vista teorico.

5.2 La Formula del Prodotto di Esponenziali

La cinematica diretta nella **forma spaziale** del Prodotto di Esponenziali afferma che la posa dell'organo terminale in una configurazione $q \in \mathbb{R}^n$ è

> **Teorema 5.1 (Formula PoE — Forma Spaziale).** La mappa di cinematica diretta $T : \mathbb{R}^n \to SE(3)$ del manipolatore è

$$T(q) = \exp(q_1\,[\xi_1]) \cdot \exp(q_2\,[\xi_2]) \cdot \ldots \cdot \exp(q_n\,[\xi_n]) \cdot T_{home} \quad (5.3)$$

Dimostrazione.

Si verifica la formula per induzione sul numero di giunti. Per $n = 1$, la rotazione di q_1 attorno all'asse ξ_1 produce la trasformazione $\exp(q_1[\xi_1])$ applicata a T_{home}, che per la definizione dell'esponenziale su SE(3) è esattamente la posa dell'organo terminale dopo una rotazione di q_1. Per il passo induttivo, si noti che $\exp(q_{i+1}[\xi_{i+1}])$ agisce a sinistra dei termini rimanenti: questo corrisponde alla composizione del moto del giunto $i+1$ espresso nello stesso frame spaziale di riferimento in cui sono definiti tutti gli screw axes. L'invarianza degli assi nella configurazione di riferimento garantisce che la composizione a sinistra abbia l'interpretazione geometrica corretta.

□

Esiste anche la **forma corporea** della formula PoE, in cui gli screw axes sono definiti nel frame dell'organo terminale nella configurazione di riferimento:

$$T(q) = T_{home} \cdot \exp(q_1\,[B_1]) \cdot \exp(q_2\,[B_2]) \cdot \ldots \cdot \exp(q_n\,[B_n]) \quad (5.4)$$

con $B_i = Ad_{T^{-1}_{home}}\,\xi_i$. Le due forme sono correlate dalla relazione di coniugio tra gli screw axes spaziali e corporei.

5.3 Lo Jacobiano Spaziale tramite la Mappa Aggiunta

Lo **Jacobiano spaziale** $J_s(q) \in \mathbb{R}^{6xn}$ lega le velocità articolari alla velocità corporea dell'organo terminale espressa nel frame di base. La sua derivazione sistematica tramite la mappa Ad è il contributo teorico più significativo del formalismo PoE.

Si differenzia la forma spaziale (5.3) rispetto al tempo. Definendo le matrici parziali $T_i(q) = \exp(q_1[\xi_1]) \cdots \exp(q_i[\xi_i])$ (con $T_0 = I$), si ha

$$\dot{T} = \Sigma_i \, T_{i-1} \cdot [\xi_i] \cdot \dot{q}_i \cdot \exp(q_i[\xi_i]) \cdot T_{i+1} \cdots \exp(q_n[\xi_n]) \, T_{home} \quad (5.5)$$

La velocità corporea dell'organo terminale $V = T^{-1}\dot{T} \in se(3)$ si ottiene moltiplicando a sinistra per T^{-1}. Per la *i*-esima colonna, il contributo è

$$T^{-1} \dot{T}|_{col\ i} = [Ad_{T\ i-1}(q) \, \xi_i] \, \dot{q}_i \quad (5.6)$$

donde la struttura dello Jacobiano spaziale:

> **Teorema 5.2 (Jacobiano Spaziale via Ad).** La *i*-esima colonna dello Jacobiano spaziale $J_s(q)$ è
>
> $$J_{s,i}(q) = Ad_{Ti-1}(q) \, \xi_i \; = \; Ad_{\exp(q_1[\xi_1]) \cdots \exp(q_{i-1}[\xi_{i-1}])} \, \xi_i \quad (5.7)$$

Dimostrazione.

Si calcola la derivata della forma PoE (5.3) rispetto a q_i. Usando la regola del prodotto per matrici e il fatto che $\partial/\partial q_i \exp(q_i[\xi_i]) = [\xi_i] \exp(q_i[\xi_i])$ (derivata a sinistra dell'esponenziale su un gruppo di Lie):

$$\partial T/\partial q_i = T_{i-1}(q) \cdot [\xi_i] \cdot \exp(q_i[\xi_i]) \cdot T_{i+1}(q) \cdots \exp(q_n[\xi_n]) \, T_{home}$$

Moltiplicando a sinistra per $T(q)^{-1}$, il torsore corrispondente nel frame corporeo è

$$T^{-1} \, (\partial T/\partial q_i) = T^{-1} \, T_{i-1} \, [\xi_i] \, (T_{i-1})^{-1} \, T = Ad_{Ti-1} \, [\xi_i] \in se(3)$$

Il vettore coordinato di questo torsore è proprio $Ad_{Ti-1(q)} \, \xi_i$, che è la *i*-esima colonna di $J_s(q)$.

□

Questa derivazione rivela la struttura profonda dello Jacobiano: la colonna *i*-esima è lo screw axis ξ_i del giunto *i trasportato* al frame corrente tramite la composizione delle trasformazioni aggiunta dei giunti precedenti. Cambiando configurazione, ciascuna colonna ruota e trasla di conseguenza, ma la formula rimane la stessa — non è mai necessario ridefinire i sistemi di riferimento dei link come nella convenzione DH. Lo Jacobiano corporeo $J_b(q)$, riferito al frame dell'organo terminale, è correlato a quello spaziale da $J_b(q) = Ad_{T(q)}^{-1} J_s(q)$.

5.4 Analisi SVD dell'Ellissoide di Manipolabilità

La decomposizione a valori singolari (SVD) dello Jacobiano fornisce una caratterizzazione geometrica completa della capacità di movimento dell'organo terminale in una data configurazione. Sia $J(q) \in \mathbb{R}^{6x\mathbf{n}}$ lo Jacobiano completo (spaziale o geometrico). La sua SVD è

$$J(q) = U \Sigma V^T, \quad U \in \mathbb{R}^{6x6}, \Sigma \in \mathbb{R}^{6x\mathbf{n}}, V \in \mathbb{R}^{\mathbf{n}x\mathbf{n}} \quad (5.8)$$

dove U e V sono ortogonali e $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$ con $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ e $r = \text{rank}(J)$. Le colonne di U sono le direzioni principali dello spazio operativo e le colonne di V le direzioni principali dello spazio articolare.

Si considera il **insieme delle velocità operative** raggiungibili con velocità articolari di norma unitaria: $\mathcal{W}(q) = \{\dot{x} = J(q)\,\dot{q} : \|\dot{q}\| \leq 1\}$. Questo insieme è un ellissoide in $\mathbb{R}^6$ — l'**ellissoide di manipolabilità** — con semi-assi orientati lungo le colonne di U e lunghezze pari ai valori singolari σ_i. In formula:

$$\mathcal{W}(q) = \{\, \dot{x} \in \mathbb{R}^6 : \dot{x}^T (J J^T)^{-1} \dot{x} \leq 1 \,\} \quad (5.9)$$

La matrice di manipolabilità $W(q) = J(q)\,J^T(q) \in \mathbb{R}^{6x6}$ è simmetrica semidefinita positiva, e la sua decomposizione spettrale $W = \Sigma_{i=1}^{r} \sigma_i^2\, u_i u_i^T$ coincide con quella ottenuta via SVD. L'indice scalare di Yoshikawa è

$$w(q) = \sqrt{(\det W(q))} = \sigma_1 \, \sigma_2 \cdots \sigma_r \quad \text{(prodotto dei valori singolari non nulli)} \tag{5.10}$$

che si annulla quando J perde rango, ossia quando il robot si trova in una singolarità. Un indicatore più robusto della qualità della configurazione è il **condition number** $\kappa(J) = \sigma_1/\sigma_r$, che misura l'isotropia dell'ellissoide: $\kappa = 1$ corrisponde a un'ellissoide sferica (direzioni tutte ugualmente raggiungibili) e $\kappa \to \infty$ alla singolarità.

Per la risoluzione di problemi di ottimizzazione della configurazione, è utile il gradiente dell'indice di manipolabilità rispetto alle coordinate articolari. Si può dimostrare che

$$\partial w/\partial q = w(q) \cdot \mathrm{vec}^{\top}(J^{\dagger\top}) \cdot \partial(\mathrm{vec}\, J)/\partial q \tag{5.11}$$

dove vec(·) denota l'operatore di vettorizzazione per colonne e $J^{\dagger}$ è lo pseudo-inverso di J. Questa formula consente di calcolare il gradiente di w rispetto a q con un costo $O(n^2)$, sfruttando la struttura dello Jacobiano del manipolatore.

5.5 Ottimizzazione nel Null-Space per Robot Ridondanti

Per manipolatori ridondanti ($n > 6$), esiste un null-space di J(q) di dimensione $d = n - r \geq 1$. I moti articolari nel null-space non producono effetti sulla velocità dell'organo terminale ma modificano la configurazione del robot, e possono quindi essere impiegati per ottimizzare secondari criteri di prestazione senza interferire con il task primario.

Sia $H : Q \to \mathbb{R}$ una funzione obiettivo secondaria (ad esempio $H(q) = w(q)$ per massimizzare la manipolabilità, o $H(q) = -\tfrac{1}{2}\|q - q_{center}\|^2$ per tenere il robot vicino alla configurazione centrale). La legge di ridondanza resolution che massimizza H nell'istante corrente è la **proiezione del gradiente nel null-space**:

$$\dot{q} = J^{\dagger}(q)\,\dot{x} + k\,(I - J^{\dagger}(q)\,J(q))\,\partial H/\partial q, \quad k > 0 \tag{5.12}$$

Il secondo addendo è un vettore articolare proiettato ortogonalmente nel null-space di J, cosicché la velocità dell'organo terminale è sempre $\dot{x} = J(q)\,\dot{q}$ come richiesto. Questa soluzione locale può tuttavia convergere a massimi locali o produrre fenomeni di ciclismo (path-dependent final configuration) quando il task è ciclico. Una formulazione più corretta dal punto di vista globale si ottiene tramite l'ottimizzazione su tutta la traiettoria, che richiede la soluzione di un problema variazionale.

La condizione necessaria di ottimalità per la legge (5.12) si ricava dalla teoria delle proiezioni: il vettore $(I - J^{\dagger}J)\,\partial H/\partial q$ è la componente di $\partial H/\partial q$ ortogonale all'immagine di $J^{\top}$, ovvero il gradiente ristretto alla varietà $N(J) = \ker(J)$. Quando la matrice J non ha rango pieno (singolarità), il null-space si estende discontinuamente, fenomeno che può causare instabilità nella legge di controllo (5.12) e giustifica l'impiego dello pseudo-inverso smorzato anziché di quello esatto.

Ottimizzazione quadratica con vincoli di velocità articolare

In presenza di vincoli di diseguaglianza sulle velocità articolari $-\dot{q}_{\min} \le \dot{q} \le \dot{q}_{\max}$ — il problema di ridondanza resolution si formula come un programma quadratico (QP) ad ogni istante:

$$\min \;\; \tfrac{1}{2}\,\|\dot{q} - \dot{q}_0\|^2_W \tag{5.13}$$

$$\text{s.t. } J(q)\,\dot{q} = \dot{x}_d, \quad \dot{q}_{\min} \le \dot{q} \le \dot{q}_{\max}$$

dove $\dot{q}_0$ è un vettore di riferimento nel null-space (ad esempio il gradiente di H) e $\|\cdot\|_W$ è la norma ponderata con una matrice W diagonale definita positiva che riflette l'importanza relativa dei giunti. Il QP ha 2n vincoli lineari e n variabili, ed è risolvibile con algoritmi di punto interno o di active set con complessità $O(n^3)$ nel caso peggiore, ma generalmente molto più veloce sfruttando la struttura sparsa.

5.6 Ottimizzazione Riemanniana su SO(3) e SE(3)

I capitoli precedenti hanno trattato l'ottimizzazione di funzioni definite su spazi euclideei $\mathbb{R}^n$. Numerosi problemi della robotica richiedono invece di ottimizzare rispetto a variabili che vivono direttamente su SO(3) o SE(3): la calibrazione estrinseca delle telecamere, la registrazione di point cloud (ICP, NDT), l'inizializzazione degli encoder nei world model latenti e la stima della posa iniziale in Dreamer sono tutti problemi in cui l'incognita è un elemento di SE(3). Applicare la discesa del gradiente euclidea ignorando la curvatura della varietà produce soluzioni che violano i vincoli di ortogonalità e determinante, costringendo a proiezioni correttive ad hoc. L'ottimizzazione Riemanniana risolve il problema generalizzando la discesa del gradiente alla geometria intrinseca della varietà.

Gradiente Riemanniano e spazio tangente

Sia $f : SO(3) \to \mathbb{R}$ una funzione liscia da minimizzare. Lo spazio tangente a SO(3) nel punto R è $T_R SO(3) = \{R\Omega : \Omega \in so(3)\} = \{R\Omega : \Omega^T = -\Omega\}$, isomorfo a $\mathbb{R}^3$ tramite la hat map. Il gradiente euclideo $\nabla f(R) \in \mathbb{R}^{3\times3}$ non appartiene in generale a $T_R SO(3)$, poiché è una matrice generica. Il gradiente Riemanniano $\text{grad } f(R) \in T_R SO(3)$ si ottiene proiettando il gradiente euclideo sullo spazio tangente:

$$\text{grad } f(R) = R\,\text{skew}(R^T \nabla f(R)) \quad \text{con} \quad \text{skew}(A) = \tfrac{1}{2}(A - A^T) \tag{5.14}$$

La formula (5.14) si verifica osservando che, per la metrica bi-invariante di SO(3) $\langle A, B\rangle_R = \text{tr}(A^T B)$, il gradiente Riemanniano è l'unico elemento di $T_R SO(3)$ tale che $\langle \text{grad } f(R), \xi\rangle_R = Df(R)[\xi]$ per ogni $\xi \in T_R SO(3)$. Sostituendo la definizione del prodotto scalare e sviluppando la derivata direzionale, si ottiene $\text{skew}(R^T \nabla f)$ come coordinata nell'algebra so(3).

Retrazione tramite mappa esponenziale e discesa del gradiente su SO(3)

Per restare sulla varietà dopo un passo di discesa, si usa la retrazione R : TSO(3) → SO(3). La scelta naturale per i gruppi di Lie è la mappa esponenziale: dato il vettore di passo $\eta \in so(3)$, la retrazione è

$$R_{k+1} = R_k \cdot \exp(-\alpha\, [\text{grad } f(R_k)]^{\vee}) \quad (5.15)$$

dove $\alpha > 0$ è il passo, $[\cdot]^{\vee}$ è la matrice antisimmetrica corrispondente al gradiente Riemanniano, e l'esponenziale matriciale si calcola con la formula di Rodrigues (1.4). La moltiplicazione a destra $R_k \cdot \exp(\cdot)$ garantisce che $R_{k+1} \in SO(3)$ per ogni valore di α, eliminando la necessità di ri-ortogonalizzazione. Per f L-smooth rispetto alla metrica di SO(3), la discesa del gradiente Riemanniano (5.15) con $\alpha \leq 1/L$ converge a un punto critico con tasso O(1/k), in piena analogia con il Teorema 12.1 per le funzioni euclidee. La generalizzazione a SE(3) sostituisce il gradiente Riemanniano con il bitorsore $\text{grad } f(T) \in T^{b}SE(3)$ e la retrazione con $T_{k+1} = T_k \cdot \exp(-\alpha\, [\text{grad } f(T_k)]^{\wedge})$, dove il cap-operator è la mappa da se(3) a matrici omogenee 4×4 (formula 1.9).

Applicazioni: calibrazione estrinseca e ICP differenziabile

Nella calibrazione estrinseca di una telecamera rigidamente montata sul robot, si vuole stimare $T_e = (R_e, t_e) \in SE(3)$ che mappa il frame del sensore nel frame del link. Dati N punti di corrispondenza 3D-3D (p^I, q^I) con p^I nel frame della telecamera e q^I nel frame del robot, la funzione di costo è $f(T_e) = \sum^I \| T_e p^I - q^I \|^2$. Il gradiente Riemanniano e l'iterazione (5.15) su SE(3) convergono al minimo globale (funzione convessa per punti non coplanari) in $O(\log(1/\varepsilon))$ iterazioni, senza mai uscire da SE(3). Lo stesso framework si applica all'ICP differenziabile: la stima della trasformazione ottima $T_k \in SE(3)$ che allinea due point cloud è un'iterazione alternante tra il passo di corrispondenza (nearest-neighbour) e il passo di ottimizzazione Riemanniana su SE(3), che produce aggiornamenti $T_{k+1} = T_k \cdot \exp(-\alpha \text{ grad } f(T_k))$ garantiti di restare in SE(3) ad ogni iterazione.

Capitolo 6 — Controllo Geometrico su Varietà di Lie: Lee, Leok e McClamroch

Il controllo dei moti rigidi su SO(3) e SE(3) pone sfide fondamentali che le tecniche di controllo classico, sviluppate per spazi euclideei, non sono in grado di affrontare in modo soddisfacente. Quando si utilizza una rappresentazione minimale di SO(3) (come gli angoli di Eulero), le singolarità della parametrizzazione rendono impossibile una legge di controllo globale: il controllore deve inevitabilmente presentare discontinuità o comportamenti degeneri in alcune configurazioni. La soluzione proposta da T. Lee, M. Leok e N. H. McClamroch nel lavoro del 2010 su IEEE TAC è costruire le leggi di controllo *direttamente su SO(3) e SE(3)*, senza mai coordinatizzare le varietà, e dimostrare la stabilità **quasi-globale** — stabile ovunque tranne che in un insieme di misura nulla.

6.1 Funzione di Lyapunov Geometrica su SO(3)

Si considera il problema di tracking di orientamento: dato un orientamento desiderato $R_d(t) \in SO(3)$ con velocità angolare desiderata $\Omega_d(t) \in \mathbb{R}^3$ (nel frame corporeo), si vuole progettare una coppia $M \in \mathbb{R}^3$ tale che $R(t) \to R_d(t)$ asintoticamente. Il corpo rigido segue la dinamica

$$\dot{R} = R\,[\Omega]^\times, \quad J\,\Omega = M - \Omega \times J\,\Omega \tag{6.1}$$

dove $J \in \mathbb{R}^{3\times3}$ è il tensore di inerzia (costante nel frame corporeo) e $\Omega \in \mathbb{R}^3$ è la velocità angolare corporea. La **funzione di Lyapunov geometrica** $\Psi : SO(3) \to [0, 2]$ è definita come

> **Definizione 6.1 (Funzione Ψ su SO(3)).** Per $R, R_d \in SO(3)$, la funzione di errore configurazionale è

$$\Psi(R, Rd) = \tfrac{1}{2}\,\mathrm{tr}(I - R_d^{\top} R) \tag{6.2}$$

Le proprietà fondamentali di Ψ sono:

(i) $\Psi(R, R_d) \geq 0$ con $\Psi = 0$ se e soltanto se $R = R_d$ (configurazione desiderata raggiunta).

(ii) $\Psi(R, R_d) = 2$ quando $R = -R_d$ (i.e., $\mathrm{tr}(R_d^T R) = -3$), configurazione diametralmente opposta.

(iii) $\Psi(R, R_d) = 1 - \cos(\theta/2)^2$, dove θ è l'angolo dell'asse di rotazione da R_d a R, quindi Ψ è un minorante del quadrato della distanza geodetica.

Dimostrazione.

Si dimostra la proprietà (iii). Sia $R_e = R_d^T R \in SO(3)$ la matrice di errore; allora $\Psi = \frac{1}{2}\,\mathrm{tr}(I - R_e)$. Per la formula di Rodrigues, $R_e = I + \sin\theta\,[k]^\times + (1 - \cos\theta)[k]^2$. Si calcola $\mathrm{tr}(R_e) = \mathrm{tr}(I) + \sin\theta\,\mathrm{tr}([k]^\times) + (1 - \cos\theta)\,\mathrm{tr}([k]^2) = 3 + 0 + (1 - \cos\theta)(\|k\|^2 - 3) = 3 + (1 - \cos\theta)(1 - 3) = 1 + 2\cos\theta$. Dunque $\Psi = \frac{1}{2}(3 - (1 + 2\cos\theta)) = 1 - \cos\theta = 2\sin^2(\theta/2)$, e per l'identità $\sin^2(\theta/2) = (1 - \cos(\theta/2)^2)/\ldots$ (via half-angle formula) si ottiene $\Psi = 2\sin^2(\theta/2)$, che è effettivamente ≥ 0 con minimo in $\theta = 0$ ($R = R_d$) e massimo $\Psi = 2$ in $\theta = \pi$ ($R = -R_d$ nel caso $\hat{k}$ arbitrario).

□

6.2 Il Vettore di Errore sull'Algebra di Lie

Per costruire la legge di controllo, è necessario definire un vettore di errore nello spazio euclideo $\mathbb{R}^3$ che rappresenti la "distanza" di R da R_d in SO(3). Introducendo la matrice di errore $R_e = R_d^T R$, il vettore di errore di orientamento $\mathbf{e}_R \in \mathbb{R}^3$ è definito tramite l'operatore di vee (*vee map*) $(\cdot)^\vee : so(3) \to \mathbb{R}^3$, inverso della hat map:

$$e_R = \tfrac{1}{2}\,(R_d^T R - R^T R_d)^\vee = \tfrac{1}{2}\,(R_e - R_e^T)^\vee \quad (6.3)$$

Il vettore e_R è lo vettore assiale della parte antisimmetrica di R_e, ed è la coordinata dell'elemento di so(3) più vicino a $R_e - I$ (approssimazione al primo ordine del logaritmo di R_e per angoli piccoli). Le sue proprietà sono:

$\|e_R\| = \sin\theta$ (dove θ è l'angolo di R_e); per θ piccolo, $e_R \approx \theta\,\hat{k}$ (asse-angolo di R_e); $\partial\Psi/\partial R = e_R$ (gradiente di Ψ rispetto alla configurazione su SO(3)).

Il vettore di errore sulla velocità angolare è

$$e\Omega = \Omega - R^T R_d\,\Omega_d \quad \in \mathbb{R}^3 \quad (6.4)$$

ossia la differenza tra la velocità angolare corporea corrente e quella desiderata, entrambe trasportate nel frame corporeo corrente. La dinamica dell'errore (R_e, e_Ω) si ottiene differenziando (6.3) e (6.4):

$$\dot{R}e = Re\,[e\Omega]^\times, \quad J\,\dot{e}\Omega = M - e\Omega \times J\Omega - J(\hat{\Omega}\,R^T R_d\,\Omega_d - R^T R_d\,\dot{\Omega}_d) \quad (6.5)$$

6.3 La Legge di Controllo di Lee-Leok-McClamroch

> **Teorema 6.1 (Legge di Controllo su SO(3)).** La legge di controllo

$$M = -k_R\,e_R - k_\Omega\,e\Omega + \Omega \times J\Omega - J(\hat{\Omega}\,R^T R_d\,\Omega_d - R^T R_d\,\dot{\Omega}_d) \quad (6.6)$$

con costanti $k_R, k_\Omega > 0$, rende l'equilibrio $(R_e, e_\Omega) = (I, 0)$ **quasi-globalmente asintoticamente stabile**: l'insieme degli stati iniziali per cui la traiettoria non converge all'equilibrio ha misura di Lebesgue nulla (è la varietà instabile dell'equilibrio indesiderato $R_e = -R_d$).

Dimostrazione.

Si sceglie la funzione candidata di Lyapunov

$$V(Re, e\Omega) = k_R\,\Psi(Re, I) + \tfrac{1}{2}\,e\Omega^T J\,e\Omega \quad \geq 0 \quad (6.7)$$

dove si è omesso l'argomento R_d per semplicità. La derivata temporale di Ψ lungo le traiettorie è

$$\dot{\Psi} = \tfrac{1}{2}\,\mathrm{tr}(-\dot{R}_d^T R - R_d^T \dot{R}) = \tfrac{1}{2}\,\mathrm{tr}(-R_d^T R\,[e\Omega]^\times) = e_R \cdot e\Omega \quad (6.8)$$

dove si è usata la proprietà $\mathrm{tr}(A\,[\omega]^\times) = -2\,a^\vee \cdot \omega$ per A antisimmetrica con $a^\vee$ il suo vettore assiale, e si è riconosciuta la definizione di e_R come parte antisimmetrica di R_e. La derivata di $\tfrac{1}{2}\,e_\Omega^T J e_\Omega$ lungo le traiettorie della dinamica in anello chiuso è

$$½\ d/dt\ (e\Omega^{\top}\ J\ e\Omega) = e\Omega^{\top}\ J\ \dot{e}\Omega$$

Sostituendo la dinamica (6.5) e la legge di controllo (6.6):

$$J\ \dot{e}\Omega = -kR\ eR - k\Omega\ e\Omega + \Omega \times J\Omega - J(\hat{\Omega}\ R^{\top}Rd\ \Omega d - R^{\top}Rd\ \dot{\Omega}d)$$

$$-\ (e\Omega \times J\ \Omega + J\ (\hat{\Omega}\ R^{\top}\ Rd\ \Omega d - R^{\top}\ Rd\ \dot{\Omega}d))$$

$$= -kR\ eR - k\Omega\ e\Omega$$

I termini di compensazione feed-forward (Ω $R^{\top}Rd\ \Omega d$ − $R^{\top}Rd\ \dot{\Omega}d$) si cancellano esattamente con i corrispondenti termini nella (6.5). Quindi

$$e\Omega^{\top}\ J\ \dot{e}\Omega = -kR\ eR^{\top}\ e\Omega - k\Omega\ \|e\Omega\|^2$$

Sommando le derivate dei due termini di V:

$$\dot{V} = kR\ eR \cdot e\Omega - kR\ eR^{\top}\ e\Omega - k\Omega\ \|e\Omega\|^2\ =\ -k\Omega\ \|e\Omega\|^2\ \leq\ 0 \quad (6.9)$$

La funzione V è non crescente. Poiché $\dot{V} = 0$ implica $e_{\Omega} = 0$ (velocità angolare di errore nulla), e sostituendo $e_{\Omega} \equiv 0$ nella dinamica dell'errore si ottiene $\dot{e}_{\Omega} = -(k_R/J)e_R = 0$, dunque $e_R = 0$, cioè $R_e = I$. Per il principio di invarianza di LaSalle, la traiettoria converge all'insieme invariante più grande contenuto in $\{\dot{V} = 0\}$, che è l'unione degli equilibri: $R_e = I$ (equilibrio desiderato) e $R_e = -R_d$ (equilibri indesiderati con R_e tale che $tr(R_e) = -3$). L'equilibrio indesiderato è instabile per la scelta di Ψ come funzione di Lyapunov, e il suo bacino di attrazione è esattamente la varietà instabile, che ha codimensione 1 in $SO(3) \times \mathbb{R}^3$ e quindi misura nulla. Pertanto, per quasi ogni condizione iniziale, la traiettoria converge a $(R_e, e_{\Omega}) = (I, 0)$.

□

La condizione di quasi-globalità è ottimale nel senso topologico: non può esistere una legge di controllo continua globalmente stabile su SO(3), perché SO(3) è una varietà compatta con gruppo fondamentale non triviale, e il teorema di Borsuk-Ulam implica che qualsiasi campo vettoriale continuo su SO(3) deve avere almeno un zero — che

corrisponde a un ulteriore punto di equilibrio. La quasi-globalità è il meglio che si possa ottenere con un controllo continuo.

6.4 Analisi della Stabilità Locale e Linearizzazione

Per angoli di errore piccoli ($\|\theta\| \ll 1$, con θ l'angolo di R_e), si ha $e_R \approx \theta\,\hat{k} \approx \tfrac{1}{2}(R_e - R_e^{\top})^{\vee}$ e la dinamica dell'errore si linearizza come

$$J\ddot{\theta} + k_\Omega J\dot{\theta} + k_R\theta \approx 0 \quad \text{(sistema lineare disaccoppiato)} \tag{6.10}$$

dove $\theta \in \mathbb{R}^3$ è il vettore asse-angolo dell'errore. Il sistema (6.10) è un oscillatore del secondo ordine con frequenza naturale $\omega_n^2 = k_R/J$ (prendendo J scalare per semplicità) e coefficiente di smorzamento $\zeta = k_\Omega/(2\omega_n)$. La risposta al gradino e il tempo di assestamento sono quindi controllabili indipendentemente dalle direzioni del tensore J scegliendo appropriatamente il rapporto k_R/k_Ω.

6.5 Estensione all'Intero Gruppo SE(3)

L'estensione del controllo geometrico all'intero gruppo SE(3) richiede la formulazione congiunta degli errori di orientamento e posizione. Sia $T = (R, p) \in SE(3)$ la posa corrente del corpo rigido con origine al baricentro, $T_d(t) = (R_d, p_d)$ la posa desiderata. La matrice di errore è $T_e = T_d^{-1}T = (R_e, p_e)$ con $R_e = R_d^{\top}R$ e $p_e = R_d^{\top}(p - p_d)$.

La **funzione di Lyapunov geometrica su SE(3)** estende naturalmente Ψ aggiungendo il termine di errore posizionale:

$$V_{SE(3)}(T_e, e_V) = k_R\,\Psi(R_e, I) + \tfrac{1}{2}k_p\,\|p_e\|^2 + \tfrac{1}{2}\,e_V^{\top}\,M\,e_V \tag{6.11}$$

dove $M \in \mathbb{R}^{6\times 6}$ è la matrice di inerzia corporea del solido (includendo sia massa che inerzia rotazionale) ed $e_V \in \mathbb{R}^6$ è il torsore di errore di velocità nel frame corporeo:

$$e_V = V - \mathrm{Ad}_{T_e^{-1}}\,V_d \quad \in \mathbb{R}^6 \tag{6.12}$$

con $V = (v, \Omega)$ la velocità corporea corrente e V_d quella desiderata. La legge di controllo su SE(3) ha la struttura

$$F = -k_p\, e_p - k_R\, e_R - k_d\, e_v + ad_{V}^{*}(M\,V) - M\,(ad_{V_e}\,Ad_{T_e}^{-1}\,V_d - Ad_{T_e}^{-1}\,\dot{V}_d) \quad (6.13)$$

dove $F \in \mathbb{R}^6$ è il bitorsore (wrench) di controllo, $e_p = p_e$ e e_R sono i vettori di errore posizionale e orientazionale, $e_v = e_V$ il vettore di errore di velocità corporea, e ad_V^* è la mappa co-aggiunta piccola. Il termine $-M\,ad_{V_e}\,Ad_{T_e}^{-1}\,V_d$ compensa la variazione del sistema di riferimento durante il moto, mentre $-M\,Ad_{T_e}^{-1}\,\dot{V}_d$ compensa l'accelerazione desiderata.

> **Teorema 6.2 (Stabilità quasi-globale su SE(3)).** La legge di controllo (6.13) con $k_p, k_R, k_d > 0$ adeguatamente scelti rende l'equilibrio $T_e = I$, $e_V = 0$ quasi-globalmente asintoticamente stabile. L'insieme instabile ha misura nulla in $SE(3) \times \mathbb{R}^6$.

La dimostrazione segue lo stesso schema del caso SO(3): si calcola $\dot{V}_{SE(3)}$ lungo le traiettorie, si mostra che $\dot{V}_{SE(3)} = -k_d \|e_V\|^2 \le 0$, e si applica il principio di invarianza di LaSalle. I dettagli tecnici richiedono di mostrare che il termine $\tfrac{1}{2} k_p \|p_e\|^2 + k_R \Psi(R_e, I)$ è una funzione di Lyapunov locale sul gruppo SE(3), proprietà che segue dalla convessità locale della metrica bi-invariante di SE(3) nella coordinata asse-angolo per angoli $\theta \in (-\pi, \pi)$.

Capitolo 7 — Control Barrier Functions e Sicurezza Certificabile per Sistemi Robotici

Premessa e collocazione nel percorso

I capitoli precedenti hanno sviluppato le leggi di controllo che portano un manipolatore verso una configurazione desiderata — PD con compensazione gravitazionale (§4.1), controllo a coppia calcolata (§4.2), LQR (§4.3), MPC (§4.4), controllo geometrico su $SO(3)$ e $SE(3)$ (Capitolo 6). Tutte queste tecniche rispondono alla domanda *come raggiungere un obiettivo*; nessuna risponde alla domanda altrettanto urgente *come garantire che il sistema non entri in regioni di stato pericolose durante il moto*.

Il divario non è trascurabile. Un braccio robotico che opera in prossimità di un operatore umano deve evitare collisioni con certificazione formale, non soltanto con strategie euristiche. Il §13.9 ha menzionato il conformal prediction come possibile rimedio nell'ambito dei world model, ma tale approccio presuppone già definita la nozione di insieme sicuro e non

fornisce leggi di controllo esplicite. Il presente capitolo colma questa lacuna presentando la teoria dei **Control Barrier Functions (CBF)**, che fornisce condizioni necessarie e sufficienti affinché un insieme compatto di stati sia positivamente invariante rispetto alla dinamica in anello chiuso, e il **QP CLF–CBF**, che combina stabilizzazione e sicurezza in un unico problema di ottimizzazione quadratica risolvibile in tempo reale.

Sistemi di controllo affini e notazione

Si considera la classe dei sistemi di controllo affini nell'ingresso

$$\dot{x} = f(x) + g(x)\,u, \quad x \in R^n, \quad u \in U \subseteq R^m,$$

dove $f: R^n \to R^n$ e $g: R^n \to R^{n\times m}$ sono funzioni localmente Lipschitz. Questa classe include l'equazione del manipolatore (3.3) dopo la linearizzazione ingresso–uscita (4.2), la linearizzazione attorno a un punto di equilibrio per LQR (4.4), e le dinamiche dei sistemi di locomozione in coordinate di corpo rigido. Per una funzione liscia $V: R^n \to R$, la derivata di Lie è $L_f V(x) = \nabla V(x)^\top f(x)$ e analogamente $L_g V(x) = \nabla V(x)^\top g(x) \in R^{1\times m}$. La derivata totale lungo la soluzione di (7.1) è:

$$\dot{V}(x,u) = L_f V(x) + L_g V(x)\,u.$$

Funzioni di Lyapunov di Controllo (CLF)

Definizione 1 (Control Lyapunov Function). Una funzione liscia $V: R^n \to R_{\geq 0}$ è una *funzione di Lyapunov di controllo* (CLF) per il sistema (7.1) se è definita positiva, propria (ovvero $V(x) \to \infty$ per $\|x\| \to \infty$), e soddisfa la *condizione di decrescenza controllabile*:

$$\inf_{u\in U} \left[L_f V(x) + L_g V(x)\,u\right] \leq -\gamma(V(x)), \quad \forall\, x \neq 0,$$

dove $\gamma: R_{\geq 0} \to R_{\geq 0}$ è una funzione di classe K.

La condizione (7.2) generalizza la condizione di Lyapunov classica $\dot{V} < 0$: richiede che esista *almeno un* ingresso u capace di rendere $\dot{V}$ negativa. Il legame con la stabilizzabilità è preciso.

Teorema 2 (Artstein–Sontag 1983–1989). *Il sistema (7.1) ammette una CLF liscia se e soltanto se è globalmente asintoticamente stabilizzabile.*

Dimostrazione (schema). Il verso "⇐" segue dal teorema di converse Lyapunov per sistemi stabilizzabili: se esiste un feedback $u = k(x)$ che rende l'origine GAS, la funzione di Lyapunov del sistema in anello chiuso $\dot{x} = f(x) + g(x)k(x)$ è una CLF per il sistema aperto, poiché $inf_u\left[L_f V + L_g V\, u\right] \leq L_f V(x) + L_g V(x)k(x) = \dot{V}|_{u=k(x)} \leq -\gamma(V(x))$.

Il verso "⇒", dovuto a Sontag (1989), si costruisce esplicitamente tramite la *Sontag formula*: se $a(x) = L_f V(x)$ e $b(x) = L_g V(x)^{\top}$, la legge

$$k(x) = \{ -\frac{a+\sqrt{a^2+\|b\|^4}}{\|b\|^2}\, b,\ b\neq 0,\ 0,\ b = 0,$$

è continua ovunque tranne l'origine (dove si può estendere con continuità), e soddisfa $L_f V(x) + L_g V(x)k(x) \leq -\sqrt{a^2 + \|b\|^4} \leq 0$ con uguaglianza solo se $a = b = 0$, il che, per la definita positività di V, accade soltanto in $x = 0$. ▫ □

Control Barrier Functions: definizione e condizione di invarianza

Un insieme $C \subset R^n$ è detto *sicuro* per il sistema (7.1) se ogni traiettoria che parte in C vi rimane per ogni $t \geq 0$: si dice che C è *positivamente invariante*. Si considera un insieme della forma

$$C = \{x \in R^n : h(x) \geq 0\},$$

dove $h: R^n \to R$ è una funzione liscia con $\partial C = \{x: h(x) = 0\}$ regolare ($\nabla h(x) \neq 0$ su ∂C).

Definizione 3 (Control Barrier Function, Ames et al. 2017). La funzione h che definisce (7.3) è una *Control Barrier Function (CBF)* per il sistema (7.1) se esiste una funzione di classe K estesa α tale che

$$sup_{u \in U}\left[L_f h(x) + L_g h(x)\, u + \alpha(h(x))\right] \geq 0, \quad \forall\, x \in C.$$

Il significato geometrico di (7.4) è diretto. Sul bordo ∂C, dove $h(x) = 0$ e quindi $\alpha(h(x)) = 0$, la condizione diventa $sup_u\left[L_f h + L_g h\, u\right] \geq 0$: deve esistere un ingresso che faccia puntare il campo vettoriale verso l'interno di C (o tangente al bordo), impedendo l'uscita. Nella parte interna, dove $h(x) > 0$, la funzione $\alpha(h(x)) > 0$ crea un margine di sicurezza che consente $\dot{h}$ temporaneamente negativa purché non troppo grande in valore assoluto.

Teorema 4 (Invarianza positiva tramite CBF). *Se h è una CBF per (7.1) e la legge di controllo $u = k(x)$ soddisfa*

$$L_f h(x) + L_g h(x)\, k(x) + \alpha(h(x)) \geq 0 \quad \forall\, x \in C,$$

allora C è positivamente invariante per il sistema in anello chiuso.

Proof. Si definisce la funzione ausiliaria $\varphi(t) = h(x(t))$. Lungo le soluzioni del sistema in anello chiuso,

$$\dot{\varphi}(t) = L_f h(x(t)) + L_g h(x(t))\, k(x(t)) \geq -\, \alpha(h(x(t))) = -\, \alpha(\varphi(t)).$$

Si tratta di un'inequazione differenziale scalare della forma $\dot{\varphi} \geq -\, \alpha(\varphi)$. Integrando mediante la disuguaglianza di Gronwall per funzioni di classe K estese: se $\varphi(0) = h(x(0)) \geq 0$, allora $\varphi(t) \geq \beta(\varphi(0), t) \geq 0$ per ogni $t \geq 0$, dove β è la soluzione del sistema confronto $\dot{y} = -\, \alpha(y)$, $y(0) = \varphi(0)$. Poiché α è di classe K estesa, il sistema confronto ha soluzione globale e

$y(t)\geq 0$ per ogni $t\geq 0$ (si verifica che $y = 0$ è un equilibrio non attraversabile). Dunque $h(x(t))\geq 0$ per ogni $t\geq 0$, il che significa $x(t)\in C$. ▫ □

Osservazione 5. La scelta $\alpha(s) = \alpha_0 s$ con $\alpha_0 > 0$ è la più comune nella pratica e corrisponde alla condizione lineare $\dot{h}(x)\geq -\alpha_0 h(x)$: la funzione di barriera decade al massimo esponenzialmente, ma non cambia segno. La scelta $\alpha(s) = \alpha_0 s^{1/3}$ produce barriere più aggressive (il sistema può avvicinarsi più rapidamente al bordo) ma rimane valida poiché α è di classe K estesa.

CBF di ordine superiore per vincoli relativi al grado

In molte applicazioni robotiche la funzione di sicurezza $h(x)$ non dipende direttamente dall'ingresso u: ad esempio, se si vuole imporre un limite di posizione $h(q) = q_{max} - q_i \geq 0$ per l'articolazione i-esima, il gradiente $\partial h/\partial x$ ha componente nulla sulle velocità, cosicché $L_g h \equiv 0$. In questo caso la Definizione 3 non è applicabile direttamente e occorre derivare h lungo il campo fino a far comparire l'ingresso.

Definizione 6 (CBF di ordine relativo r). Sia $r\geq 1$ il più piccolo intero tale che $L_g L_f^{r-1} h(x)\neq 0$ per qualche $x\in C$. Si definisce la sequenza di funzioni ausiliarie:

$$\psi_0(x) = h(x),\ \psi_i(x) = \dot{\psi}_{i-1}(x) + \alpha_i\left(\psi_{i-1}(x)\right), \quad i = 1, \ldots, r-1,$$

con α_i funzioni di classe K estese. La funzione h è una *High-Order CBF (HOCBF)* se:

$$sup_u\left[L_f^r h(x) + L_g L_f^{r-1} h(x)\, u + O(\alpha, \psi)\right] \geq 0 \quad \forall\, x \in \bigcap_{i=0}^{r-1} \{\psi_i(x)\geq 0\},$$

dove $O(\alpha, \psi)$ raccoglie i termini che dipendono dalle α_i e dalle ψ_{i-1}.

Esempio: limite articolare come HOCBF di ordine 2. Per il manipolatore con dinamica (7.5), $x = \left(q, \dot{q}\right)$, e il vincolo di posizione $h\left(q, \dot{q}\right) = q_{max} - q_i$. Si ha $L_g h = 0$ poiché h non dipende da $\dot{q}$, ma $\dot{h} = -\dot{q}_i$ e $\ddot{h} = -\ddot{q}_i$, che dipende da τ tramite la matrice di inerzia. Applicando la sequenza (7.6)–(7.7) con $r = 2$:

$$\psi_0 = q_{max} - q_i,\ \psi_1 = \dot{\psi}_0 + \alpha_1\left(\psi_0\right) = -\dot{q}_i + \alpha_1\left(q_{max} - q_i\right).$$

La condizione HOCBF sulla derivata seconda è:

$$-\ddot{q}_i + \dot{\alpha}_1\left(\psi_0\right) + \alpha_2\left(\psi_1\right) \geq 0.$$

Poiché $\ddot{q}_i$ dipende linearmente da τ tramite la riga i-esima di $M^{-1}(q)$, questa diventa un vincolo lineare in τ, integrabile direttamente nel QP.

Il QP CLF–CBF: formulazione e analisi

Il QP CLF–CBF, introdotto da Ames, Grizzle e Tabuada (2014) e generalizzato da Ames et al. (2019), combina in un unico problema di ottimizzazione la *stabilizzazione* verso un obiettivo (tramite CLF) e il *rispetto dei vincoli di sicurezza* (tramite CBF). Il controller nominale $u_{nom}(x)$ — calcolato con un qualsiasi metodo (es. controllo a coppia calcolata, MPC con orizzonte breve) — viene modificato minimalmente per soddisfare entrambi i requisiti:

$$u^*(x) = arg\,min_{u \in R^m}\ \frac{1}{2}\|u - u_{nom}(x)\|_W^2 + p\,\delta^2\ sogg.\ a\quad L_f V(x) + L_g V(x)\,u \leq -\gamma V(x) + \delta\quad ($$

dove $W \succ 0$ è una matrice di peso, $\delta \in R$ è una variabile di rilassamento che rende il vincolo CLF soft (priorità alla sicurezza sulla convergenza), e $p \gg 0$ è il costo del rilassamento.

Il problema (7.8) è un QP parametrico in x: ha funzione obiettivo quadratica convessa e vincoli lineari in (u, δ). È risolvibile in tempo

costante (indipendente dalla dimensione dello spazio degli stati, dipendente solo da m e dal numero di vincoli k) con algoritmi di punto interno o active-set, tipicamente in microsecondi su hardware embedded.

Teorema 7 (Ammissibilità e garanzie del QP CLF–CBF). *Assumiamo che le funzioni* $h_1, ..., h_k$ *siano CBF per (7.1) e che V sia una CLF. Allora per ogni* $x \in \bigcap_j C_j$*:*

1. *Il QP (7.8) ammette sempre una soluzione ottima* $\left(u^*, \delta^*\right)$.
2. *Le soluzioni del sistema in anello chiuso con* $u = u^*(x)$ *rimangono in* $\bigcap_j C_j$ *per ogni* $t \geq 0$.
3. *Se i vincoli CLF e CBF sono compatibili (nessun conflitto), allora* $\delta^* = 0$ *e il sistema converge asintoticamente verso* $x = 0$.

Proof. Punto (i). La funzione obiettivo è strettamente convessa in (u, δ). Poiché h_j è una CBF, per ogni $x \in C_j$ esiste almeno un u che soddisfa il vincolo CBF j-esimo (definizione 3). Il vincolo CLF ha sempre la soluzione ammissibile $\delta = L_f V + L_g V u + \gamma V$ con u qualsiasi, quindi l'insieme ammissibile è non vuoto. Poiché l'obiettivo è coercivo, il minimo è attinto.

Punto (ii). Per costruzione, $u^*(x)$ soddisfa i vincoli CBF rigidi in (7.8). Per il Teorema 4, ciascun C_j è positivamente invariante, dunque anche la loro intersezione (che è descrivibile con il vincolo congiunto $min_j h_j(x) \geq 0$) è positivamente invariante.

Punto (iii). Se non c'è conflitto, $\delta^* = 0$ e il vincolo CLF è soddisfatto strettamente: $L_f V + L_g V u^* \leq -\gamma V$. Per la Definizione 1 e il teorema di LaSalle, il sistema converge all'origine. ▫ □

Osservazione 8 (Conflitto CLF–CBF). Il conflitto tra vincoli CLF e CBF — denominato *infeasibility* — è il principale limite pratico del QP (7.8).

Accade quando l'unica direzione che riduce V viola un vincolo CBF: il sistema deve rallentare la convergenza per restare sicuro. La variabile di rilassamento δ gestisce il conflitto degradando gracefully la velocità di convergenza senza mai compromettere la sicurezza.

Estensione a sistemi con incertezza e world model

I sistemi robotici reali operano in presenza di incertezze parametriche e di perturbazioni esterne. In presenza di un termine additivo sconosciuto $\Delta(x,t)$ con $\|\Delta(x,t)\| \leq d(x)$ noto, la dinamica perturbata è:

$$\dot{x} = f(x) + g(x)\,u + \Delta(x,t).$$

Definizione 9 (Robust CBF, Jankovic 2018). La funzione h è una *Robust CBF (RCBF)* per (7.9) se esiste $\alpha \in K$ tale che

$$sup_u\left[L_f h + L_g h\,u - \|\nabla h\|\,d(x) + \alpha(h)\right] \geq 0 \quad \forall\, x \in C,$$

ovvero che la condizione di barriera tenga nel caso peggiore della perturbazione.

La connessione con i world model è diretta. Dato un world model che produce la stima dell'incertezza epistemica $U_{epi}(s,a)$ come definita nell'eq. (20.4) della monografia (varianza del disaccordo tra i modelli dell'ensemble), si può porre:

$$d(x) = \beta \cdot U_{epi}(s,a),$$

dove $\beta > 0$ è un fattore di scalatura che tiene conto della dimensione. In questo modo, il QP CLF–CBF si adatta automaticamente all'incertezza del modello: nelle regioni ben esplorate (bassa U_{epi}) il vincolo RCBF è quasi identico al CBF nominale; nelle regioni non esplorate il margine di sicurezza si allarga conservativamente.

Teorema 10 (Conformal CBF). *Sia* $\epsilon > 0$ *il livello di copertura desiderato. Dato un insieme di calibrazione di N transizioni reali* $\{(x_i, u_i, x_i')\}_{i=1}^{N}$, *si definisce il quantile di calibrazione* $q_{1-\epsilon} = Quantile_{1-\epsilon}\{\|x_i' - \hat{f}(x_i, u_i)\|\}_{i=1}^{N}$.

Se la distribuzione dei residui è scambiabile, allora con probabilità $1 - \epsilon$ *sulla prossima transizione reale:* $\|x' - \hat{f}(x,u)\| \leq q_{1-\epsilon}$,

ovvero $d(x) = q_{1-\epsilon}$ *è una stima valida per la RCBF (7.10) con copertura* $1 - \epsilon$.

Proof. Segue direttamente dalla proprietà di copertura marginale del conformal prediction (Vovk et al. 2005): per qualsiasi distribuzione scambiabile, il quantile empirico a livello $1 - \epsilon$ garantisce $P(R_{N+1} \leq q_{1-\epsilon}) \geq 1 - \epsilon$, dove $R_i = \|x_i' - \hat{f}(x_i, u_i)\|$ è il residuo della i -esima transizione. ▫ □

Applicazioni alla robotica manipolativa

Limiti articolari e di velocità

Per un manipolatore a n gradi di libertà, i vincoli di posizione $q_{min,i} \leq q_i \leq q_{max,i}$ e di velocità $|\dot{q}_i| \leq \dot{q}_{max,i}$ si traducono rispettivamente in HOCBF di ordine 2 e CBF di ordine 1. Definiamo:

$$h_{q,i}^{+}(x) = q_{max,i} - q_i,\ h_{q,i}^{-}(x) = q_i - q_{min,i},\ h_{\dot{q},i}^{+}(x) = \dot{q}_{max,i} - \dot{q}_i.$$

Le prime due richiedono la costruzione HOCBF (§6.4), le ultime sono CBF di ordine 1 dirette. Il QP CLF–CBF con tutti i $4n$ vincoli (due di posizione e due di velocità per ogni giunto) rimane un QP con $4n$ vincoli lineari, risolvibile in $O(n^3)$ nel caso peggiore ma tipicamente molto più veloce.

Evitamento di ostacoli

Per un ostacolo sferico di centro c_{obs} e raggio r_{obs}, la funzione di sicurezza basata sulla distanza euclidea dell'organo terminale $p(q)$ dall'ostacolo è:

$h_{obs}(q) = \|p(q) - c_{obs}\|^2 - r_{obs}^2$.

Il gradiente è $\nabla_q h_{obs} = 2J(q)^\top \big(p(q) - c_{obs}\big)$, dove $J(q)$ è lo Jacobiano geometrico. Questo è un CBF di ordine relativo 1 rispetto alla velocità dell'organo terminale, e ordine 2 rispetto al comando di coppia. La condizione CBF diventa:

$2\big(p - c_{obs}\big)^\top J(q)\,\dot{q} + \alpha\big(h_{obs}\big) \geq 0.$

Sostituendo $\dot{q} = J^\dagger \dot{p}_d + \big(I - J^\dagger J\big)z$ (risoluzione cinematica inversa con null-space, eq. (2.8) della monografia), si ottiene un vincolo lineare in $\dot{p}_d$ e z.

Note conclusive sul capitolo

Il QP CLF–CBF fornisce le garanzie formali di sicurezza che le tecniche puramente di apprendimento non possono fornire: l'invarianza positiva dell'insieme sicuro è certificata *per costruzione* dalla condizione CBF, indipendentemente dalla qualità del world model. L'integrazione con l'incertezza epistemica del world model — tramite la RCBF o il conformal CBF — rende il framework adattivo: più dati reali vengono raccolti, più $d(x)$ diminuisce e meno conservativo diventa il vincolo di sicurezza. Questo circolo virtuoso tra apprendimento e sicurezza è la base teorica del paradigma *safe reinforcement learning* che verrà ripreso nel Capitolo 20.

Capitolo 8 — Il Neural Tangent Kernel: Reti Neurali come Strutture Differenziali

Il Neural Tangent Kernel (NTK), introdotto da Jacot, Gabriel e Hongler nel 2018, è una costruzione che emerge dallo studio del comportamento di reti neurali profonde nel limite di larghezza infinita durante il processo di ottimizzazione con discesa del gradiente. Il risultato fondamentale è che, in questo regime, la funzione realizzata dalla rete si comporta come un processo gaussiano con kernel fisso determinato dalla struttura dell'architettura. Questo collegamento tra ottimizzazione nonlineare delle reti e teoria classica dei kernel consente di analizzare la convergenza dell'addestramento, la generalizzazione e le proprietà spettrali dell'apprendimento con strumenti algebrici e funzionali.

Il NTK assume importanza capitale nella robotica e nell'intelligenza artificiale incarnata per due ragioni convergenti: da un lato, fornisce strumenti per comprendere teoricamente la capacità di approssimazione e la velocità di convergenza dei modelli neurali che entrano come componenti nelle architetture di World Models e JEPA; dall'altro, l'analisi spettrale del NTK illumina il tipo di funzioni che una rete neurale può efficacemente apprendere, con implicazioni dirette per la scelta delle architetture.

8.1 Formulazione del Problema e Linearizzazione

Sia $f(\theta, x) : \mathbb{R}^p \times \mathbb{R}^d \to \mathbb{R}^m$ una rete neurale profonda con parametri $\theta \in \mathbb{R}^p$. Si considera l'addestramento su un dataset $S = \{(x_i, y_i)\}_{i=1}^N$ minimizzando una funzione di perdita $\mathcal{L}(\theta) = \Sigma_i \, \ell(f(\theta, x_i), y_i)$. La discesa del gradiente in tempo continuo segue la traiettoria

$$d\theta/dt = -\nabla\theta \, \mathcal{L}(\theta) = -\Sigma_i \, (\partial f(\theta, x_i)/\partial\theta)^\top \, \partial\ell/\partial f_i \quad (7.1)$$

La chiave analitica è considerare la linearizzazione di f attorno ai parametri iniziali θ_0 (regime di apprendimento *lazy*), in cui i parametri si muovono di poco rispetto alla loro inizializzazione. La rete linearizzata è

$$f_{lin}(\theta, x) = f(\theta_0, x) + J\theta(\theta_0, x)\,(\theta - \theta_0) \quad (7.2)$$

dove $J\theta(\theta_0, x) = \partial f(\theta_0, x)/\partial\theta \in \mathbb{R}^{mxp}$ è lo Jacobiano della rete rispetto ai parametri calcolato nell'inizializzazione. Nella dinamica linearizzata, la variazione dell'output della rete segue l'equazione differenziale

$$df(\theta(t), x)/dt = J\theta(\theta_0, x) \cdot d\theta/dt = -J\theta(\theta_0, x)\,\Sigma_i\, J\theta(\theta_0, x_i)^T\, r(x_i, t) \quad (7.3)$$

dove $r(x_i, t) = \partial\ell/\partial f|_{f=f(\theta(t),x_i)}$ è il residuo nella *i*-esima coppia al tempo t. Introducendo il vettore di output $\hat{u}(t) = (f(\theta(t), x_1)^T, \ldots, f(\theta(t), x_N)^T) \in \mathbb{R}^{Nm}$, l'equazione del moto della rete linearizzata diventa

$$d\hat{u}/dt = -K^{NTK}\,\partial\mathcal{L}/\partial\hat{u} \quad (7.4)$$

8.2 Il Kernel di Tangente Neurale

> **Definizione 7.1 (Neural Tangent Kernel).** Il **Neural Tangent Kernel** associato alla rete $f(\theta, \cdot)$ è la funzione $K^{NTK} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{mxm}$ definita da

$$K^{NTK}(x, x') = J\theta(\theta, x)\, J\theta(\theta, x')^T = \Sigma_{\square}\, (\partial f/\partial\theta_{\square})(x) \otimes (\partial f/\partial\theta_{\square})(x') \quad (7.5)$$

Per addestramento con perdita dei minimi quadrati $\ell(f, y) = \tfrac{1}{2}\|f - y\|^2$, la dinamica (7.4) diventa lineare in $\hat{u}$:

$$d\hat{u}/dt = -K^{NTK}\,(\hat{u} - y) \quad (7.6)$$

dove $y = (y_1^T, \ldots, y_N^T)^T \in \mathbb{R}^{Nm}$ è il vettore dei target. La soluzione è

$$\hat{u}(t) = y + e^{-K^{NTK}\,t}\,(\hat{u}(0) - y) \quad (7.7)$$

La convergenza è quindi esponenzialmente rapida se e soltanto se K^{NTK} è definita positiva. La velocità di convergenza nella direzione del *k*-esimo autovettore di K^{NTK} è determinata dall'autovalore λ_k: le direzioni con λ_k

grande convergono rapidamente (*frequenze alte*), quelle con λ_k piccolo molto lentamente (*frequenze basse*).

8.3 Il Limite di Larghezza Infinita e Convergenza al Kernel Costante

> **Teorema 7.1 (Convergenza del NTK, Jacot-Gabriel-Hongler 2018).** Sia $f(\theta, x)$ una rete fully-connected di profondità L e larghezza h, con inizializzazione gaussiana $\theta \sim N(0, \sigma^2 I/h)$. Nel limite $h \to \infty$, il NTK $K^{NTK}_h(x, x')$ converge in probabilità a un kernel deterministico $K^{\infty}(x, x')$ che rimane costante durante l'addestramento con discesa del gradiente.

Dimostrazione.

La dimostrazione procede per induzione sulla profondita L. **Base**: per L = 1, la rete f = W2 sigma(W1x + b1) + b2. Ogni contributo ha varianza O(1/h). La somma su h termini converge in probabilita a $K^{infty}(x, xp)$. **Passo induttivo**: per L strati il calcolo ricorsivo del kernel propaga il risultato tramite aspettative di funzioni di attivazione gaussiane. La costanza segue da norma-delta-theta = O(1/radice-h) che tende a zero.

□

Il Teorema 7.1 ha una conseguenza profonda: nel regime di larghezza infinita, l'addestramento di una rete neurale con discesa del gradiente è equivalente alla regressione con il kernel K^{∞}, e il predittore appreso è

$$f(\theta(\infty), x) = K^{\infty}(x, X)\,[K^{\infty}(X, X)]^{-1}\,y \quad (7.8)$$

dove $X = (x_1, \ldots, x_N)$ è la matrice dei dati di addestramento, ovvero l'esatto predittore della regressione kernel con K^{∞}. Questo collega formalmente le reti neurali alle macchine a kernel e consente di usare la ricca teoria degli spazi di Hilbert a kernel riproducente (RKHS) per analizzare le proprietà di generalizzazione.

8.4 Il NTK per l'Architettura Transformer

L'estensione del NTK ai Transformer richiede particolare attenzione a causa della struttura del meccanismo di self-attention, che è una funzione nonlineare di tipo *quadratico-esponenziale* nei dati di input. Si considera il blocco di attenzione a testa singola applicato a un input $X \in \mathbb{R}^{T\times d}$ (sequenza di T token di dimensione d):

$$\mathrm{Attn}(X) = \mathrm{softmax}(XWQ\,(XWK)^{\top} / \sqrt{d}) \cdot X\,WV \;=\; A(X) \cdot X\,WV \quad (7.9)$$

dove W_Q, W_K, $W_V \in \mathbb{R}^{d\times d}$ sono le matrici di query, key e value, e $A(X) = \mathrm{softmax}(XW_QW_K^{\top}X^{\top}/\sqrt{d}) \in \mathbb{R}^{T\times T}$ è la matrice di attenzione. Il NTK del Transformer per due sequenze $X, X' \in \mathbb{R}^{T\times d}$ ha la struttura

$$K^{NTK}_{Attn}(X, X') = \Sigma_{\theta\in\{WQ,WK,WV,Wout,\ldots\}}\ (\partial f/\partial\theta)(X) \cdot (\partial f/\partial\theta)^{\top}(X') \quad (7.10)$$

Il contributo dello strato di attenzione al NTK è particolarmente complesso perché il gradiente $\partial\mathrm{Attn}/\partial W_Q$ dipende dal prodotto $XW_QW_K^{\top}X^{\top}$, rendendo il kernel dipendente in modo non banale dalla geometria dei token nella sequenza. Un risultato chiave, dimostrato da Hron et al. (2020), afferma che anche i Transformer con proiezioni di query, key e value scalate opportunamente convergono a un processo gaussiano con kernel deterministico nel limite di larghezza infinita, a patto di mantenere fissa la profondità L.

Il kernel di attenzione in larghezza infinita ha la forma ricorsiva

$$K^{Attn,(l)}(X, X') = K^{(l-1)}(X, X') + K^{(l-1)}(X, X) \cdot K^{(l-1)}(X', X') \cdot \Lambda(X, X') \quad (7.11)$$

dove $\Lambda(X, X')$ è una matrice di correzione che dipende dalla funzione softmax e riflette l'interazione tra i pattern di attenzione sui due input. La ricorrenza (7.11) mostra che il NTK del Transformer accumula contributi da tutti gli strati, con gli strati profondi che modificano il kernel degli strati superficiali tramite il termine di correzione Λ — meccanismo che non ha analogo nelle reti fully-connected.

8.5 Decomposizione Spettrale e Bias Induttivo

La struttura spettrale del NTK determina il **bias induttivo** della rete: le funzioni con coefficienti di Fourier (o proiezioni sui autovettori del kernel) corrispondenti ad autovalori grandi vengono apprese rapidamente, mentre quelle corrispondenti ad autovalori piccoli vengono apprese lentamente o non vengono apprese affatto con la regolarizzazione implicita della discesa del gradiente.

Per un kernel traslazionalmente invariante $K(x, x') = \kappa(x - x')$ su $\mathbb{R}^d$, il teorema di Bochner afferma che κ è un kernel positivo definito se e soltanto se è la trasformata di Fourier di una misura spettrale non negativa μ

$$\kappa(x - x') = \int_{\mathbb{R}^d} e^{i\omega\cdot(x-x')}\, d\mu(\omega) \quad (7.12)$$

Gli autovalori del kernel decadono come $\text{lambda}_k \sim k^{\text{neg}2s/d}$ per kernel di regolarita s (Matern, Sobolev), governato dal teorema di Mercer. Il tasso di apprendimento del *k*-esimo pattern e

$$\|f(t) - f^*\|^2 \sim \Sigma\square\ \lambda\square^2\, e^{-2\lambda\square t}\, |\alpha\square|^2 + \Sigma\square\ |\alpha\square|^2 \quad (7.13)$$

dove i coefficienti alpha_k = inner-prod(f*, phi-k) del target f* nella base degli autovettori del kernel esprimono la spectral bias delle reti neurali: esse apprendono prima le componenti a bassa frequenza (autovalori grandi) e in seguito quelle ad alta frequenza. La spectral bias ha implicazioni dirette nell apprendimento di World Models e JEPA.

8.6 NTK, Generalizzazione e Dimensionalità Intrinseca

Il rischio di generalizzazione di un predittore kernel è controllato dalla teoria di Rademacher, che lega il gap di generalizzazione alla complessità della funzione appresa nella norma dell'RKHS associato al kernel:

$$R(f) - \hat{R}(f) \leq \|f\|_\kappa \cdot O(\sqrt{(\log(1/\delta)/N)}) \quad (7.14)$$

dove $R(f)$ è il rischio atteso, $\hat{R}(f)$ il rischio empirico e $\|f\|_K^2 = \langle f, f \rangle_K$ è la norma nell'RKHS. Il bound (7.14) mostra che la generalizzazione è buona quando la norma-K del predittore è piccola. La discesa del gradiente con loss L_2 e regolarizzazione implicita trova proprio il predittore di minima norma-K che interpola i dati, cioè la soluzione minima-norma della regressione kernel (7.8).

Questo collegamento rivela una relazione profonda con la nozione di **dimensionalità intrinseca** dello spazio di ipotesi: se il target f^* appartiene a uno spazio di Sobolev di regolarità s su $\mathbb{R}^d$, e il kernel ha la stessa regolarità, allora il tasso di convergenza ottimale è $N^{-s/(2s+d)}$, tasso che soffre della maledizione della dimensionalità — ma soltanto rispetto alla dimensione intrinseca dei dati, non alla dimensione ambiente. Per le architetture di tipo Transformer o JEPA che operano in spazi latenti di bassa dimensione intrinseca, questo implica che i tassi di convergenza effettivi possono essere ben superiori a quelli predetti dalla dimensione nominale dei dati grezzi (pixel, audio, punti nube).

8.7 Critiche e Limiti della Teoria NTK

Pur fornendo uno strumento analitico potente, la teoria NTK ha limiti fondamentali che è necessario riconoscere esplicitamente. In primo luogo, il regime di larghezza infinita con parametri che si spostano di poco dall'inizializzazione (lazy training) è lontano dal regime operativo delle reti neurali nelle applicazioni pratiche: le reti usate in robotica e visione artificiale non sono larghe al punto da rientrare nel regime del Teorema 7.1, e i loro parametri si muovono significativamente durante l'addestramento, formando le cosiddette *caratteristiche neurali* (feature learning) che non sono catturate dal NTK.

In secondo luogo, il kernel K^∞ del Transformer è tipicamente molto diverso dai kernel comunemente usati nella pratica (RBF, Matérn), e la sua struttura implicita è difficile da analizzare analiticamente. Lavori recenti

(Yang & Hu 2021, Bordelon et al. 2022) hanno esteso la teoria a reti *feature-learning* tramite il formalismo μP (maximal update parameterization) e il Tensor Program Framework, che permettono di analizzare il comportamento di reti pratiche nel limite di larghezza infinita con aggiornamenti di peso finiti. In questo framework più generale, il kernel non è costante durante l'addestramento ma evolve secondo una dynamics kernels equation che accoppia la traiettoria dei parametri alla struttura del kernel, catturando il meccanismo di apprendimento delle rappresentazioni che è alla base di JEPA e dei World Models.

8.8 Reti SE(3)-Equivarianti: dal Lemma di Schur alle architetture

La Parte Prima ha dimostrato che le configurazioni robotiche vivono su gruppi di Lie SO(3) e SE(3). Le architetture neurali destinate a elaborare dati geometrici — point cloud di scene 3D, pose di oggetti, traiettorie nello spazio dei compiti — dovrebbero incorporare questa simmetria come vincolo strutturale, anziché apprenderla empiricamente da dati. Una rete f : $\mathbb{R}^d \to \mathbb{R}^m$ è equivariante rispetto al gruppo G se, per ogni g ∈ G e ogni input x:

$$f(\rho\_in(g) \cdot x) = \rho\_out(g) \cdot f(x) \quad (7.15)$$

dove ϱ_in e ϱ_out sono rappresentazioni di G (rispettivamente sullo spazio di input e di output). Per G = SE(3), ϱ può essere la rappresentazione banale (scalari, invarianti), la rappresentazione vettoriale (vettori in $\mathbb{R}^3$ che ruotano con R), o più in generale rappresentazioni irriducibili di tipo L (armoniche sferiche di ordine L).

Fondamento: il Lemma di Schur e le rappresentazioni irriducibili di SO(3)

Le rappresentazioni irriducibili (irreps) di SO(3) sono parametrizzate dall'intero non negativo $L \in \mathbb{N}_0$ e hanno dimensione 2L+1. Per L = 0: scalari; L = 1: vettori in $\mathbb{R}^3$; L = 2: tensori simmetrici privi di traccia (5-dimensionali); e così via. Il Lemma di Schur fornisce il vincolo fondamentale che qualsiasi mappa lineare equivariante deve soddisfare:

> **Lemma (Schur, 1905).** Siano V_L e V_{L'} spazi su cui agiscono le irreps di SO(3) di ordine L e L'. Qualsiasi mappa lineare $\varphi : V_L \to V_{L'}$ equivariante rispetto a SO(3) soddisfa:
>
> $$\varphi = \{ c \cdot Id \;\; se\, L = L', \quad 0 \;\; se\, L \neq L' \} \quad (7.16)$$
>
> In altri termini, l'unica operazione lineare equivariante tra due irreps è l'identità scalata (se di stesso ordine) o la mappa nulla (se di ordine diverso).

Dimostrazione.

Sia $\varrho_L : SO(3) \to GL(V_L)$ la L-esima irrep. Per equivarianza, φ commuta con ogni elemento del gruppo: $\varphi \circ \varrho_L(g) = \varrho_{L'}(g) \circ \varphi$ per ogni $g \in SO(3)$. Se $L \neq L'$, le due irreps sono non isomorfe; per il Lemma di Schur nella sua forma standard (Fulton-Harris, 2004), qualsiasi intertwinore tra rappresentazioni irriducibili non isomorfe è necessariamente nullo: $\varphi = 0$. Se L = L', le due irreps sono isomorfe, e l'intertwinore è un multiplo dell'isomorfismo canonico, cioè $\varphi = c \cdot Id$ per $c \in \mathbb{R}$.

□

Il Lemma di Schur ha una conseguenza diretta sulla costruzione di reti equivarianti: qualsiasi strato lineare equivariante deve separare i canali per tipo (ordine L) e può soltanto scalare ciascun tipo indipendentemente. La nonlinearità deve essere implementata in modo equivariante — tipicamente tramite il prodotto tensoriale di Clebsch-Gordan.

Il prodotto tensoriale di Clebsch-Gordan

Il prodotto di due irreps V_{L_1} e V_{L_2} si decompone come somma diretta di irreps tramite i coefficienti di Clebsch-Gordan $C^{L_3}_{m_3;m_1m_2}$:

$$V_{L_1} \otimes V_{L_2} = \bigoplus_{L_3 = |L_1 - L_2|}^{L_1 + L_2} V_{L_3} \quad (7.17)$$

Per due feature $x^{(L_1)} \in \mathbb{R}^{2L_1+1}$ e $y^{(L_2)} \in \mathbb{R}^{2L_2+1}$, il prodotto tensoriale equivariante che produce una feature di tipo L_3 è:

$$(x \otimes_{CG} y)^{(L_3)}_m = \Sigma_{m_1, m_2} C^{L_3}_{m; m_1m_2} x^{(L_1)}_{m_1} y^{(L_2)}_{m_2} \quad (7.18)$$

Questa operazione è l'analogo equivariante della moltiplicazione element-wise nelle reti standard. Il suo costo computazionale è $O((2L_{max}+1)^6)$ per il calcolo dei coefficienti, ma nella pratica $L_{max} \leq 3$–5 rende l'operazione fattibile. Le architetture che la adottano includono e3nn (Geiger & Smidt, 2022), NequIP (Batzner et al. 2022) e MACE (Batatia et al. 2022).

SE(3)-Transformer e applicazioni alla robotica

Fuchs et al. (2020) propongono il SE(3)-Transformer, un meccanismo di attenzione che estende l'attenzione standard alle feature equivarianti di SO(3). Il kernel di attenzione tra il nodo sorgente j e il nodo destinazione i è definito come:

$$\alpha_{ij} \propto exp(\varphi_q(h_i)^{(0)} \cdot \varphi_k(h_j, r_{ij})^{(0)} / \sqrt{d}) \quad (7.19)$$

dove $r_{ij} \in \mathbb{R}^3$ è il vettore posizione relativo tra i nodi i e j, h_i è la feature del nodo i (un vettore di irreps), φ_q e φ_k sono reti lineari equivarianti che producono la query e la chiave di tipo L = 0 (scalari), garantendo che il peso di attenzione α_{ij} sia invariante per rotazioni. Il valore aggregato è:

$$v_i^{(L)} = \Sigma_j \alpha_{ij} \cdot R_L(\hat{r}_{ij}) \varphi_v^{(L)}(h_j) \quad (7.20)$$

dove R_L(r̂_{ij}) è la matrice di Wigner-D applicata alla direzione r̂_{ij} = r_{ij}/‖r_{ij}‖, che trasforma la feature del valore dal frame del nodo j al frame globale equivariante. In robotica, il SE(3)-Transformer è impiegato per: (i) la predizione della posa 6-DOF di oggetti da point cloud (FoundPose, 2024); (ii) la stima dello stato dello scheletro del robot da misure IMU con equivarianza garantita rispetto al frame di base (Ordonez-Apraez et al. 2023); (iii) il planning di presa dexterous con feature equivarianti che codificano la geometria locale dell'oggetto in modo invariante alla sua orientazione (Huang et al. 2023). In tutti questi casi, l'equivarianza garantisce che se l'input viene ruotato di $R \in SO(3)$, l'output si trasformi coerentemente senza dover reapprendere la simmetria dai dati.

Equivarianza e complessità campionaria: riduzione di un fattore $|G|$

Il vantaggio pratico dell'equivarianza non è soltanto qualitativo: si traduce in un miglioramento quantificabile della complessità campionaria, ovvero del numero di esempi di addestramento necessari per raggiungere un dato errore di generalizzazione. Il risultato seguente formalizza questa riduzione nel linguaggio della teoria della complessità di Rademacher.

> **Proposizione 7.1 (Riduzione della complessità campionaria per reti equivarianti).** Sia G un gruppo compatto che agisce su $\mathbb{R}^{\mathcal{A}}$ con misura di Haar $\mu_{\mathcal{G}}$. Sia $\mathcal{H}$ una classe di funzioni L-Lipschitz su $\mathbb{R}^{\mathcal{A}}$ e $\mathcal{H}^{\mathcal{G}} \subseteq \mathcal{H}$ la sottoclasse di funzioni equivarianti rispetto a G. La complessità di Rademacher empirica di $\mathcal{H}^{\mathcal{G}}$ su m campioni soddisfa:
>
> $$\mathfrak{R}_m(\mathcal{H}^{\mathcal{G}}) \leq \mathfrak{R}_m(\mathcal{H}) / \sqrt{|G|} \quad (7.21)$$

dove |G| è la misura di Haar del gruppo (volume di SO(3) con la metrica bi-invariante, ÷ per gruppi finiti il numero di elementi).

Dimostrazione. Per ogni $f \in \mathcal{H}_{\mathcal{G}}$ e ogni campione $(x_1, \ldots, x_{\mathcal{D}})$, si sostituisce ogni x^I con la sua media di gruppo $\dot{x}^I = \int_{\mathcal{G}} \varrho(g)x^I \, d\mu_{\mathcal{G}}(g)$. Per l'equivarianza, $f(\varrho(g)x^I) = \varrho_0(\bar{R}^t(g))f(x^I)$, quindi $f(\dot{x}^I) = \int_{\mathcal{G}} f(\varrho(g)x^I) \, d\mu_{\mathcal{G}}(g)$. Per la disuguaglianza di Jensen sulla norma euclidea e per la limitatezza L-Lipschitz di f, $\| f(\dot{x}^I) - f(\dot{x}^{I'}) \| \leq L \| \dot{x}^I - \dot{x}^{I'} \|$. La media di gruppo riduce la varianza di un fattore $|G|$ rispetto ai campioni originali (in quanto $\dot{x}^I$ è la media di $|G|$ rotazioni i.i.d. di x^I), producendo la riduzione della complessità di Rademacher dichiarata dalla disuguaglianza di Khintchine applicata alla somma di $|G|$ termini identicamente distribuiti. □

La Proposizione 7.1 ha implicazioni concrete: per SO(3) con la misura di Haar normalizzata $|G| = 8\pi^2$, la riduzione della complessità di Rademacher è di un fattore $\sqrt{(8\pi^2)} \approx 8.9$. Combinando con il bound di Rademacher (7.14), si ottiene che una rete SE(3)-equivariante addestrata su N esempi raggiunge lo stesso errore di generalizzazione di una rete non equivariante addestrata su circa 80N esempi, a parità di tutte le altre condizioni. Questa è la giustificazione teorica quantitativa del vantaggio empirico sistematicamente osservato nelle architetture equivarianti su dataset robotici: NequIP raggiunge l'accuratezza di modelli non equivarianti con 10–100× meno dati di addestramento nella predizione delle forze interatomiche, e il SE(3)-Transformer supera i Transformer standard sulla stima della posa con frazioni delle dimostrazioni. La riduzione è particolarmente preziosa nella robotica, dove raccogliere dati fisici è costoso e il numero di dimostrazioni disponibili è tipicamente nell'ordine delle centinaia anziché delle centinaia di migliaia.

Capitolo 9 — Stima Bayesiana dello Stato su Varietà di Lie: dal Filtro di Kalman al Posterior ammortizzato

Premessa e collocazione nel percorso

L'equazione del manipolatore (3.3) descrive la dinamica del sistema in termini di coordinate generalizzate $(q, \dot{q})$, assunte misurabili con precisione. Nella pratica, le misure provengono da sensori soggetti a rumore: encoder angolari con risoluzione finita, accelerometri IMU con deriva e bias, telecamere con occlusioni e rumore di quantizzazione. Il problema di ricavare da tali misure rumorose la stima ottimale dello stato del sistema — e in particolare della configurazione $T \in SE(3)$ del robot o di un oggetto nell'ambiente — è il problema della *stima bayesiana dello stato*.

La Parte Terza (world model) ha introdotto il Recurrent State Space Model (RSSM, §11.4) e ne ha motivato la struttura come architettura capace di mantenere un posterior sugli stati latenti della scena. Tuttavia, il legame formale tra il posterior ammortizzato $q_\theta(s_t | h_t, x_t)$ dell'RSSM e la teoria

classica del filtraggio bayesiano non è mai stato esplicitato. Il presente capitolo colma questa lacuna in tre stadi: si derivano il filtro di Kalman lineare, l'EKF su R^n, il Filtro di Kalman Invariante (IEKF) direttamente su $SO(3)$ e $SE(3)$, il filtro a particelle come approssimazione Monte Carlo del posterior esatto, e infine si mostra che il posterior ammortizzato dell'RSSM è una approssimazione variazionale del filtro bayesiano esatto, con il gap di approssimazione quantificato dalla divergenza KL tra il posterior vero e quello ammortizzato.

Il filtro bayesiano ricorsivo

Si considera un sistema dinamico stocastico in tempo discreto

$$x_{t+1} = f(x_t, u_t) + w_t, \quad w_t \sim p(w),\ y_t = h(x_t) + v_t, \quad v_t \sim p(v),$$

dove $x_t \in X$ è lo stato, $u_t \in U$ l'ingresso, $y_t \in Y$ l'osservazione, e w_t, v_t sono rumori di processo e di misura statisticamente indipendenti. Il problema del filtraggio bayesiano consiste nel calcolare in modo ricorsivo il *posterior filtrato*

$$p(x_t | y_{1:t}) \propto p(y_t | x_t)\, p(x_t | y_{1:t-1}),$$

a partire dal *posterior predetto*

$$p(x_t | y_{1:t-1}) = \int p(x_t | x_{t-1})\, p(x_{t-1} | y_{1:t-1})\, dx_{t-1}.$$

Queste due equazioni sono note come *step di predizione* e *step di aggiornamento* del filtro bayesiano esatto. Per sistemi lineari gaussiani, il filtro ha forma chiusa ed è il filtro di Kalman.

Filtro di Kalman: derivazione MMSE completa

Si considera il sistema lineare

$$x_{t+1} = A_t x_t + B_t u_t + w_t, \quad w_t \sim N(0, Q_t),\ y_t = C_t x_t + v_t, \quad v_t \sim N(0, R_t),$$

con stato iniziale $x_0 \sim N\left(\hat{x}_0, P_0\right)$ e i rumori reciprocamente indipendenti. Poiché la distribuzione di un sistema lineare eccitato da rumore gaussiano rimane gaussiana, il posterior è completamente caratterizzato da media e covarianza: $p\left(x_t | y_{1:t}\right) = N\left(x_t; \hat{x}_{t|t}, P_{t|t}\right)$.

Step di predizione

Dalla linearità e dalla gaussianità:

$$\hat{x}_{t|t-1} = A_{t-1}\hat{x}_{t-1|t-1} + B_{t-1}u_{t-1},\ P_{t|t-1} = A_{t-1}P_{t-1|t-1}A_{t-1}^{\top} + Q_{t-1}.$$

Step di aggiornamento

Si definisce l'innovazione $e_t = y_t - C_t\hat{x}_{t|t-1}$ con covarianza $S_t = C_t P_{t|t-1} C_t^{\top} + R_t$. Il guadagno di Kalman ottimale nel senso dei minimi quadrati medi (MMSE) è

$$K_t = P_{t|t-1} C_t^{\top} S_t^{-1}.$$

La media e covarianza aggiornate sono:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t e_t,\ P_{t|t} = \left(I - K_t C_t\right) P_{t|t-1}.$$

Teorema 11 (Ottimalità MMSE del Filtro di Kalman). *L'estimatore* $\hat{x}_{t|t}$ *definito da (9.1)–(9.2) minimizza l'errore quadratico medio* $E\left[\| x_t - \hat{x} \|^2 | y_{1:t}\right]$ *su tutti gli stimatori lineari di* x_t *dati* $y_{1:t}$.

Proof. Si considera la famiglia degli stimatori lineari $\hat{x} = L_t\tilde{x}_{t|t-1} + M_t y_t$ con matrici libere L_t, M_t. La condizione di non distorsione richiede $E\left[\hat{x}\right] = E\left[x_t | y_{1:t}\right]$, il che impone $L_t = I - M_t C_t$. L'errore di stima è $\tilde{x} = x_t - \hat{x} = \left(I - M_t C_t\right)\tilde{x}_{t|t-1} - M_t v_t$, con covarianza

$P(M_t) = (I - M_t C_t) P_{t|t-1} (I - M_t C_t)^\top + M_t R_t M_t^\top$. Minimizzando $tr[P(M_t)]$ rispetto a M_t:

$$\frac{\partial tr[P]}{\partial M_t} = -2(I - M_t C_t) P_{t|t-1} C_t^\top + 2 M_t R_t = 0,$$

da cui $M_t = P_{t|t-1} C_t^\top \left(C_t P_{t|t-1} C_t^\top + R_t \right)^{-1} = K_t$. Sostituendo si ottiene $P(K_t) = (I - K_t C_t) P_{t|t-1}$, che coincide con (9.2). ▫ □

Equazione di Riccati e stabilità del filtro

Definendo la covarianza stazionaria $P_\infty = lim_{t\to\infty} P_{t|t'}$ questa soddisfa l'*equazione algebrica di Riccati del filtraggio* (DARE):

$$P_\infty = A P_\infty A^\top + Q - A P_\infty C^\top \left(C P_\infty C^\top + R \right)^{-1} C P_\infty A^\top.$$

L'esistenza e l'unicità di $P_\infty \succ 0$ sono garantite sotto le condizioni di osservabilità di (C, A) e rilevabilità di $\left(A, Q^{1/2}\right)$, che sono le condizioni duali di quelle che garantiscono l'LQR (eq. (4.6) della monografia). La dualità filtro–regolatore è formale: l'equazione di Riccati dell'LQR (9.3) diventa quella del filtro (9.4) sostituendo $(A, B, Q, R) \to \left(A^\top, C^\top, Q, R\right)$.

EKF: linearizzazione per sistemi nonlineari

Per il sistema nonlineare (9.5)–(9.6) con $x \in R^n$, l'*Extended Kalman Filter (EKF)* approssima il posterior con una gaussiana propagata attraverso le linearizzazioni di primo ordine di f e h nell'intorno della stima corrente:

$$F_t = \frac{\partial f}{\partial x}\Big|_{x=\hat{x}_{t|t}}, \quad H_t = \frac{\partial h}{\partial x}\Big|_{x=\hat{x}_{t|t-1}}.$$

Le equazioni del filtro mantengono la struttura del Kalman lineare con $A_t \leftarrow F_t$ e $C_t \leftarrow H_t$. L'EKF è consistente (il posterior è correttamente

gaussiano) solo se le nonlinearità sono trascurabili nell'intorno della stima, condizione che può non valere per grandi angoli di rotazione.

Osservazione 12 (Limite dell'EKF per rotazioni). Per la configurazione di orientamento $R \in SO(3)$, la rappresentazione minimale con angoli di Eulero $\theta \in R^3$ è singolare nelle configurazioni canoniche (gimbal lock). Il Jacobiano $F_t = \partial f / \partial \theta$ diventa mal condizionato in tali configurazioni, rendendo l'EKF instabile. Il Filtro di Kalman Invariante (§7.5) risolve questo problema lavorando direttamente su $SO(3)$.

Il Filtro di Kalman Invariante (IEKF) su Lie Groups

Il concetto di filtro invariante fu introdotto da Bonnabel (2007) e sistematizzato da Barrau e Bonnabel (2017, 2018). L'idea portante è sostituire l'errore euclideo $\tilde{x} = x - \hat{x}$ con un *errore di gruppo* che vive nell'algebra di Lie, in modo che la sua propagazione sia lineare rispetto all'errore — indipendentemente dalla stima corrente.

Errore di gruppo e invarianza

Per un gruppo di Lie G con algebra g, si definiscono due nozioni di errore:

$$\eta_t^L = \hat{X}_t^{-1} X_t \in G \quad (\textit{errore invariante a sinistra}), \quad \eta_t^R = X_t \hat{X}_t^{-1} \in G \quad (\textit{errore invariante a destra}$$

dove $X_t \in G$ è lo stato vero e $\hat{X}_t$ è la stima. Per errori piccoli, si scrive $\eta_t \approx Exp(\varepsilon_t)$ con $\varepsilon_t \in g$, e il filtro lavora direttamente su $\varepsilon_t \in R^{dimg}$.

IEKF su $SO(3)$

Si considera la stima dell'orientamento di un corpo rigido con $X_t = R_t \in SO(3)$. Il modello di processo è il sistema di rotazione guidato dalla velocità angolare misurata con rumore:

$$R_{t+1} = R_t \, Exp(\omega_t \Delta t + w_t), \quad w_t \sim N(0, Q\,\Delta t),$$

dove $\omega_t \in R^3$ è la velocità angolare misurata dall'IMU (nel corpo) e $w_t \in R^3$ il rumore associato. Il modello di osservazione è la misurazione di un vettore di riferimento $r \in R^3$ nel frame di base (es. la gravità):

$$y_t = R_t^\top r + v_t, \quad v_t \sim N\left(0, R_{obs}\right).$$

L'errore invariante a sinistra è $\eta_t = \hat{R}_t^{-1} R_t = \hat{R}_t^\top R_t \in SO(3)$. Per errori piccoli $\eta_t \approx I + \left[\varepsilon_t\right]^\times$, la propagazione dell'errore è:

$$\varepsilon_{t+1} \approx \varepsilon_t + w_t.$$

Teorema 13 (Proprietà di log-linearità dell'IEKF su $SO(3)$). *Per il sistema (9.7)–(9.8), le equazioni di propagazione dell'errore (9.9) sono esatte al primo ordine in ε_t e indipendenti dalla stima corrente $\hat{R}_t$.*

Proof. Si calcola la propagazione dell'errore a sinistra. Per l'errore nello step di predizione:

$$\eta_{t+1}^- = \hat{R}_{t+1}^\top R_{t+1} = (\hat{R}_t Exp(\omega_t \Delta t))^\top R_t Exp(\omega_t \Delta t + w_t) = Exp(-\omega_t \Delta t) \hat{R}_t^\top R_t Exp(\omega_t \Delta t + w_t)$$

Per errori piccoli $\eta_t = Exp(\varepsilon_t) \approx I + \left[\varepsilon_t\right]^\times$, si usa la proprietà dell'aggiunta (eq. (1.11) della monografia):

$$Exp(-\omega_t \Delta t)\, Exp(\varepsilon_t)\, Exp(\omega_t \Delta t) \approx Exp\left(Ad_{Exp(-\omega_t \Delta t)} \varepsilon_t\right) \approx Exp\left(\varepsilon_t + O\left(\|\varepsilon_t\| \|\omega_t\| \Delta t\right)\right).$$

Al primo ordine in ε_t, la dipendenza da $\hat{R}_t$ scompare e si ottiene $\varepsilon_{t+1}^- \approx \varepsilon_t + w_t$, confermando (9.9). ▫ □

La proprietà di log-linearità implica che la matrice di transizione dell'errore è l'identità (al primo ordine), e la covarianza predetta è semplicemente $P_{t+1}^- = P_t + Q\,\Delta t$. Questo elimina il calcolo del Jacobiano di

transizione, rendendo l'IEKF più stabile e più efficiente dell'EKF per la stima dell'orientamento.

IEKF su $SE(3)$: stima della posa completa

Per la stima della posa del robot $X_t = T_t = \left(R_t, p_t\right) \in SE(3)$, si modella lo stato come un elemento del gruppo con il processo di locomozione in coordinate di corpo:

$$T_{t+1} = T_t \cdot Exp\left(\xi_t \Delta t + w_t\right), \quad w_t \sim N(0, \Sigma \Delta t),$$

dove $\xi_t = \left(v_t, \omega_t\right) \in R^6$ è il torsore cinematico misurato dall'odometria inerziale e $w_t \in R^6$ il rumore. Le osservazioni possono essere: misure di posizione assoluta (GPS) $y_t^p = p_t + v_t^p$, o misure di landmark $y_t^l = R_t^\top \left(l_j - p_t\right) + v_t^l$ con landmark l_j noto nel frame di base.

L'errore invariante a sinistra su $SE(3)$ è $\eta_t = T_t^{-1} \hat{T}_t$, con la coordinata in $se(3)$ data da $\varepsilon_t = \left(\varepsilon_v, \varepsilon_\omega\right) \in R^6$. Il passo di aggiornamento dell'IEKF segue lo schema del Kalman lineare sul vettore ε_t:

$$K_t = P_t^- H_t^\top \left(H_t P_t^- H_t^\top + R_t\right)^{-1}, \; \varepsilon_t = K_t \left(y_t - \hat{y}_t\right), \; \hat{T}_{t|t} = \hat{T}_{t|t-1} \cdot Exp\left(-\varepsilon_t\right), \; P_{t|t} = \left(I - K_t H_t\right) P_t^-.$$

Il termine $\hat{T} \cdot Exp\left(-\varepsilon_t\right)$ in (9.10) applica la correzione *direttamente su* $SE(3)$, senza mai uscire dal gruppo: la stima aggiornata soddisfa automaticamente il vincolo $R_{t|t} \in SO(3)$ (cioè $R^\top R = I$ e $detR = 1$), senza ri-ortogonalizzazioni a posteriori come invece richiederebbe l'EKF euclideo.

La matrice di osservazione H_t per il landmark l_j si ottiene linearizzando l'osservazione $R^\top \left(l_j - p\right)$ rispetto a $\varepsilon = \left(\varepsilon_v, \varepsilon_\omega\right)$:

$$H_t^{l_j} = \left[- R_t^\top \left[R_t^\top \left(l_j - p_t \right) \right]^\times \right] \in R^{3\times 6}.$$

Questa matrice dipende dalla posa corrente $\hat{T}_{t'}$ ma non dall'errore $\varepsilon_{t'}$ confermando che la linearizzazione è esatta al primo ordine nell'errore per il modello di landmark.

Filtro a particelle (Sequential Monte Carlo)

Quando la distribuzione di stato non è approssimabile con una gaussiana — ad esempio in presenza di multimodalità (un oggetto che può essere a sinistra o a destra) o di rumore di misura non gaussiano — il filtro a particelle approssima il posterior con un insieme ponderato di campioni (*particelle*) $\{\left(x_t^{(i)}, w_t^{(i)} \right)\}_{i=1}^N$:

$$p\left(x_t | y_{1:t} \right) \approx \sum_{i=1}^{N} w_t^{(i)} \delta_{x_t^{(i)}}\left(x_t \right), \quad \sum_i w_t^{(i)} = 1.$$

Sequential Importance Resampling (SIR)

L'algoritmo SIR, detto anche *Bootstrap Particle Filter* (Gordon, Salmond, Smith 1993), opera con la distribuzione di proposta $q\left(x_t | x_{t-1}^{(i)}, u_{t-1} \right) = p\left(x_t | x_{t-1}^{(i)}, u_{t-1} \right)$ (prior come proposta):

1. **Campionamento:** Per $i = 1, ..., N$, campionare $\tilde{x}_t^{(i)} \sim p\left(x_t | x_{t-1}^{(i)}, u_{t-1} \right)$ dal modello di processo.
2. **Pesatura:** Aggiornare i pesi con la likelihood: $\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \cdot p\left(y_t | \tilde{x}_t^{(i)} \right)$, poi normalizzare.
3. **Ricampionamento:** Se l'*effective sample size* $N_{eff} = 1 / \sum_i \left(w_t^{(i)} \right)^2$ scende sotto una soglia (tipicamente $N/2$), campionare N particelle

con reimmissione dalla distribuzione discreta $\{w_t^{(i)}\}$ e reimpostare tutti i pesi a $1/N$.

La complessità per passo è $O(N)$; la qualità dell'approssimazione cresce come $O\left(1/\sqrt{N}\right)$ per il teorema di convergenza del metodo Monte Carlo sequenziale (Del Moral 2004).

Il posterior ammortizzato dell'RSSM come filtro variazionale

Si è ora in grado di stabilire formalmente il rapporto tra il filtro bayesiano esatto e il posterior ammortizzato dell'RSSM.

Il filtro bayesiano come caso limite

Il modello generativo dell'RSSM si fattorizza secondo la eq. (11.56) della monografia:

$$p_\theta\left(x_{1:T}, s_{1:T}\right) = p\left(s_1\right)\prod_{t=2}^{T} p_\theta\left(s_t|h_t\right)\prod_{t=1}^{T} p_\theta\left(x_t|s_t\right),$$

dove $h_t = GRU\left(h_{t-1}, s_{t-1}, a_{t-1}\right)$ è lo stato deterministico ricorrente. Il posterior filtrato esatto sarebbe:

$$p\left(s_t|x_{1:t}, a_{1:t-1}\right) = \frac{p_\theta\left(x_t|s_t\right)p\left(s_t|x_{1:t-1}, a_{1:t-1}\right)}{p\left(x_t|x_{1:t-1}, a_{1:t-1}\right)},$$

dove il denominatore (evidence) è intrattabile.

L'encoder ammortizzato $q_\theta\left(s_t|h_t, x_t\right)$ approssima direttamente il posterior (9.11) senza propagare il posterior completo. Il gap di approssimazione è misurato dalla KL:

$$KL\left[q_\theta\left(s_t|h_t, x_t\right) \parallel p\left(s_t|x_{1:t}, a_{1:t-1}\right)\right] \geq 0,$$

con uguaglianza se e soltanto se l'encoder ammortizzato coincide esattamente con il posterior vero. La somma di questi gap su tutti i passi

temporali è il *gap di amortizzazione*, e la perdita totale dell'RSSM (eq. (11.38) della monografia) ne è una decomposizione:

$$\log p_\theta(x_{1:T}) = ELBO + \sum_{t=1}^{T} KL\left[q_\theta(s_t|h_t, x_t) \parallel p(s_t|x_{1:t}, a_{1:t-1})\right].$$

Teorema 14 (RSSM come filtro variazionale). *L'addestramento del modello RSSM tramite massimizzazione dell'ELBO (9.12) è equivalente a minimizzare simultaneamente:*

1. *il gap di amortizzazione (9.13) su tutti i passi;*
2. *la log-verosimiglianza negativa delle osservazioni* $-\log p_\theta(x_t|s_t)$*;*
3. *la KL tra il posterior ammortizzato e il prior dinamico* $KL\left[q_\theta(\cdot|h_t, x_t) \parallel p_\theta(\cdot|h_t)\right]$*, che impone che il prior sia un buon predittore del posterior.*

Proof. Si parte dall'identità $\log p_\theta(x_{1:T}) = ELBO + \sum_t KL[\cdots]$ dimostrata sopra, e si decompone l'ELBO secondo la eq. (11.58)–(11.59):

$$ELBO = \sum_t E_q\left[\log p_\theta(x_t|s_t)\right] - \sum_t E_q\left[KL\left(q \| p_\theta(\cdot|h_t)\right)\right].$$

Il termine $-\sum_t KL[\cdots]$ include sia la divergenza dal prior dinamico (punto (iii)) sia il gap di amortizzazione; la loro somma è esattamente $\log p_\theta(x_{1:T}) - ELBO \geq 0$. Massimizzare l'ELBO rispetto a θ minimizza quindi l'intero secondo membro. ▫ □

Il filtro di Kalman come limite lineare–gaussiano

Teorema 15 (Equivalenza IEKF – RSSM nel limite lineare–gaussiano). *Sia il modello RSSM lineare nei parametri e le distribuzioni gaussiane. Se il prior dinamico è* $p_\theta(s_t|h_t) = N(Ah_t, Q)$ *e l'encoder ammortizzato è*

$q_\theta\left(s_t|h_t, x_t\right) = N\left(\hat{s}_{t|t}, P_{t|t}\right)$, *allora il passo di aggiornamento variazionale che massimizza l'ELBO è equivalente al passo di aggiornamento del filtro di Kalman (9.14)–(9.2), con media* $\hat{s}_{t|t} = \hat{s}_{t|t-1} + K_t e_t$ *e covarianza* $P_{t|t} = \left(I - K_t C\right)P_{t|t-1}$, *dove* K_t *è il guadagno di Kalman ottimale.*

Proof. Nel caso gaussiano, la KL $KL\left[N\left(\mu_q, \Sigma_q\right) \| N\left(\mu_p, \Sigma_p\right)\right]$ ha la forma chiusa (eq. (12.64) della monografia). Il passo di aggiornamento dell'encoder ammortizzato che massimizza l'ELBO gaussiano è

$$\max_{\hat{s},P} \left[E_q\left[logp\left(x_t|s_t\right)\right] - KL[q\|p]\right].$$

Sviluppando esplicitamente (log-likelihood gaussiana meno KL gaussiana) e ottimizzando rispetto a $\hat{s}$ e P, si ottiene il sistema lineare $\partial/\partial\hat{s} = 0$, $\partial/\partial P = 0$, le cui soluzioni sono esattamente le equazioni del filtro di Kalman (9.15)–(9.2). La dimostrazione è formalmente identica alla derivazione MMSE del Teorema 11, con l'aggiunta del termine KL che gioca il ruolo della regolarizzazione verso il prior. ▫ □

Il Teorema 15 stabilisce formalmente che il filtro di Kalman è un caso speciale del posterior ammortizzato dell'RSSM nel limite lineare–gaussiano. Di converso, l'RSSM con encoder non-lineare profondo e stati categoriali (DreamerV2–V3) è una generalizzazione fortemente nonlineare del filtro bayesiano, capace di catturare distribuzioni multimodali e strutture di correlazione complesse che il filtro gaussiano non può rappresentare.

Connessione con l'IEKF su $SE(3)$

Nei task di localizzazione robotica (*SLAM*) o stima della posa di oggetti, la struttura di gruppo di $SE(3)$ deve essere preservata sia nel filtro che nel world model. L'IEKF su $SE(3)$ garantisce questa proprietà mediante le equazioni (9.16)–(9.17). Un world model RSSM con spazio latente

$s_t \in SE(3)$ — dove il prior dinamico opera con la mappa esponenziale (eq. (1.10) della monografia) e la misura di discrepanza nella KL è la distanza di Riemannian su $SE(3)$ — realizza la controparte apprendibile dell'IEKF, con la stessa garanzia strutturale di restare nel gruppo. Questa architettura è alla base dei sistemi di SLAM neurale differenziabile (CUT3R, §20.6) e dei world model per la stima della posa 6-DoF (FoundPose, §7.8).

Note conclusive sul capitolo

Il percorso da questo capitolo verso le Parti III e IV della monografia è ora formalmente connesso. Il filtro di Kalman (§7.3) è l'estimatore bayesiano ottimale per sistemi lineari gaussiani; il filtro di Kalman invariante (§7.5) ne è l'estensione naturale ai gruppi di Lie $SO(3)$ e $SE(3)$, preservando la struttura geometrica. Il filtro a particelle (§7.6) è l'approssimazione non gaussiana generale. Il posterior ammortizzato dell'RSSM (§7.7) è un'approssimazione variazionale che, nel limite lineare–gaussiano, recupera esattamente il filtro di Kalman (Teorema 15), e nella sua versione profonda e discreta (DreamerV2–V3) ne è la generalizzazione nonlineare.

Questa catena formale — filtro di Kalman → IEKF su $SE(3)$ → posterior ammortizzato RSSM — è il ponte concettuale che la monografia necessitava per rendere esplicito il filo che unisce la geometria differenziale della Parte Prima con la probabilità variazionale della Parte Terza.

Note Conclusive alla Parte Prima

I nove capitoli della Parte Prima hanno tracciato un percorso coerente che parte dalla struttura algebrica dei moti rigidi — i gruppi di Lie SO(3) e SE(3) — e arriva all'analisi differenziale delle architetture neurali tramite il Neural Tangent Kernel. Prima di procedere alle Parti successive, è utile sottolineare i fili che attraversano l'intera costruzione.

La struttura di gruppo di Lie, introdotta nel Capitolo 1, non è soltanto il contesto in cui vivono le configurazioni robotiche: essa determina attivamente come si propagano le informazioni (torsori, bitorsori) attraverso la catena cinematica (formula PoE, Capitolo 5), come si misura l'errore (funzione Ψ, Capitolo 6) e come si costruiscono leggi di controllo invarianti (Lee-Leok-McClamroch, Capitolo 6). La *equivarianza* rispetto alle simmetrie di SE(3) sarà un requisito che le architetture neurali descritte nelle Parti successive (World Models, JEPA) dovranno rispettare per acquisire rappresentazioni fisicamente coerenti.

L'equazione del manipolatore (Capitolo 3) e il controllo a coppia calcolata (Capitolo 4) hanno rivelato la struttura porto-hamiltoniana del sistema meccanico: passività, conservazione dell'energia e Jacobiano come interfaccia tra spazio articolare e spazio operativo. Il NTK (Capitolo 7) ha mostrato che reti neurali di larghezza infinita si comportano come processi gaussiani con kernel fisso, e che la loro capacità di generalizzazione è governata dalla struttura spettrale del kernel — struttura che, per i Transformer, riflette il meccanismo di attenzione e la geometria dei token nella sequenza. Questo legame tra geometria e apprendimento è la premessa concettuale per le rappresentazioni latenti di World Models e JEPA.

La Parte Seconda approfondirà i modelli neurali per la percezione e la rappresentazione, prima di procedere alle architetture che integrano previsione temporale (World Models) e apprendimento

auto-supervisionato in spazi di embedding (JEPA) — gli strumenti con cui la robotica moderna cerca di costruire agenti in grado di pianificare, ragionare e agire nel mondo fisico.

PARTE II

Fondamenti Matematici del Deep Learning per la Robotica

Capitoli 10 · 11 · 12 · 13 · 14 · 15

Capitolo 10 — Reti Neurali come Approssimatori Universali

Una rete neurale profonda è, in prima istanza, una macchina per costruire funzioni. Prima di esaminarne le proprietà statistiche e computazionali, occorre rispondere a una domanda fondamentale di natura analitica: quali famiglie di funzioni sono accessibili a tale macchina? La risposta, articolata nel corso di tre decenni di ricerca, rivela che le architetture neurali sono densamente distribuite in spazi funzionali vastissimi, il che le rende strumenti di approssimazione di portata eccezionale. Il presente capitolo affronta questa questione con il rigore degli spazi funzionali, passando dalla teoria classica degli approssimatori universali shallow alle stime quantitative di Barron, alle proprietà di separazione in profondità, fino alle implicazioni per la teoria statistica dell'apprendimento.

10.1 Impostazione formale

Sia n la dimensione dello spazio di ingresso e sia K $\subset \mathbb{R}^n$ un insieme compatto. Una rete neurale feed-forward a uno strato nascosto di larghezza N con funzione di attivazione $\sigma : \mathbb{R} \to \mathbb{R}$ realizza funzioni della forma

$$f_\theta(x) = \sum_{j=1}^{N} cj \;\; \sigma(wj^{\mathrm{T}} x + bj)$$

dove θ = {(c_j, w_j, b_j)}_{j=1}^N sono i parametri, w_j $\in \mathbb{R}^n$ i pesi, b_j $\in \mathbb{R}$ le soglie, c_j $\in \mathbb{R}$ i coefficienti di uscita. Denotiamo Σ(σ, n) l'insieme di tutte le tali reti al variare di N e dei parametri. Per reti profonde a L strati, la funzione si scrive ricorsivamente come

$$h_0(x) = x, \quad h_\ell(x) = \sigma(W_\ell h_{\ell-1}(x) + b_\ell), \quad \ell = 1, \ldots, L$$

con W_ℓ ∈ ℝ^{d_ℓ × d_{ℓ−1}} e b_ℓ ∈ ℝ^{d_ℓ}. L'uscita è f_θ(x) = W_{L+1} h_L(x) + b_{L+1}. Le funzioni di attivazione di interesse includono la sigmoide σ(t) = 1/(1+e^{−t}), la tangente iperbolica tanh(t), la ReLU(t) = max(0, t), la GELU(t) = t·Φ(t) dove Φ è la CDF gaussiana standard, e la Swish(t) = t·σ(βt).

10.2 Il Teorema di Cybenko e la densità delle reti shallow

Definizione 8.1 (Funzione discriminatoria). *Una funzione $\sigma : \mathbb{R} \to \mathbb{R}$ è discriminatoria se l'unica misura di Borel μ su $[0,1]^n$ per cui $\int \sigma(w^T x + b) d\mu(x) = 0$ per ogni $w \in \mathbb{R}^n$, $b \in \mathbb{R}$ è la misura nulla.*

Teorema 8.1 (Cybenko, 1989). *Sia $\sigma : \mathbb{R} \to \mathbb{R}$ continua e discriminatoria. Allora $\Sigma(\sigma, n)$ è denso in $C([0,1]^n)$ rispetto alla norma uniforme $\|\cdot\|_\infty$.*

La dimostrazione si appoggia al teorema di Hahn-Banach in forma duale. Supponiamo per assurdo che Σ(σ, n) non sia denso; allora la sua chiusura S è un sottospazio chiuso proprio di C([0,1]ⁿ). Per il teorema di separazione di Hahn-Banach, esiste un funzionale lineare continuo G su C([0,1]ⁿ), non nullo, che si annulla su S. Per il teorema di rappresentazione di Riesz, G è rappresentato da una misura di Borel μ su [0,1]ⁿ tale che G(h) = ∫h dμ. In particolare

$$\int [0,1]^n \sigma(w^T x + b)\, d\mu(x) = 0 \quad \forall\, w \in \mathbb{R}^n,\ \forall\, b \in \mathbb{R}$$

Per la proprietà discriminatoria di σ, questo forza μ = 0, contraddicendo G ≠ 0. L'assunzione che Σ(σ, n) non sia denso è dunque falsa. ∎

Il teorema di Hornik (1991) rimuove l'ipotesi discriminatoria e generalizza agli spazi L^p. La chiave tecnica è dimostrare che ogni funzione continua non costante e non polinomiale è discriminatoria nel senso esteso: ciò si ottiene osservando che la trasformata di Fourier della misura ∫σ(wᵀx +

b)dμ(x) = 0 si annulla su un insieme denso, da cui μ = 0 per teorema di unicità della trasformata.

Teorema 8.2 (Hornik, 1991). *Sia σ : ℝ → ℝ continua, non costante, non polinomiale. Allora Σ(σ, n) è denso in C(K) per ogni compatto K ⊂ ℝ^n, e denso in L^p(μ) per ogni misura di probabilità finita μ su ℝ^n e ogni p ∈ [1, ∞).*

10.3 Spazi funzionali rilevanti

Spazi di Sobolev W^{k,p}(Ω). Per Ω ⊂ ℝ^n aperto e k ∈ ℕ, p ∈ [1,∞), lo spazio di Sobolev è

$$W^{k,p}(\Omega) = \{f \in L\hat{}p(\Omega) : D\alpha\ f \in L\hat{}p(\Omega)\ \forall\ |\alpha| \le k\}, \quad \|f\|_{W\hat{}\{k,p\}} = \left(\sum_{|\alpha|\le k} \|D\alpha\ f\|p^p_{L\hat{}p}\right)^{1/p}$$

dove D^α denota la derivata distribuzionale di multiindice α. Gli spazi di Sobolev interpolano tra regolarità differenziale e integrabilità, e il numero di derivate k governa la velocità di approssimazione ottimale.

Spazio di Barron B_s(ℝ^n). Per s > 0, lo spazio di Barron è

$$B_s(\mathbb{R}^n) = \{f : C\_f := \int \|\omega\|s\ |f(\omega)|\, d\omega < \infty\}$$

dove f̂(ω) = ∫f(x) e^{−2πiω^Tx} dx è la trasformata di Fourier. La quantità C_f misura la distribuzione dell'energia spettrale sulle alte frequenze: funzioni con C_f < ∞ sono analiticamente regolari in senso spettrale. La classe B_1 è quella fondamentale per il teorema di Barron.

Spazi a nucleo riproducente (RKHS). Uno spazio di Hilbert H di funzioni su $\mathcal{X}$ è un RKHS se esiste K : $\mathcal{X}$ × $\mathcal{X}$ → ℝ tale che (i) K(·, x) ∈ H per ogni x, e (ii) f(x) = ⟨f, K(·, x)⟩_H per ogni f ∈ H (proprietà riproducente). Il nucleo arco-coseno per reti ReLU è definito ricorsivamente:

$$K^{(\ell+1)}(x,\, x') \;=\; \frac{\sigma^2\_w}{2\pi}\; J\;^{(1)}(\theta(\ell)\;)\;\|x\|\;\|x'\|$$

dove J^{(1)}(θ) = sin θ + (π − θ) cos θ e θ^{(ℓ)} = arccos(K^{(ℓ)}(x,x') / (√K^{(ℓ)}(x,x) √K^{(ℓ)}(x',x'))). Nel limite di larghezza infinita, questo nucleo descrive esattamente la funzione di covarianza del processo gaussiano corrispondente alla rete deep.

10.4 Il Teorema di Barron: stima quantitativa

Teorema 8.3 (Barron, 1993). *Sia $f \in B_1(\mathbb{R}^n)$ con $C\_f = \int \|\omega\|\,|\hat{f}(\omega)|\,d\omega < \infty$. Per ogni distribuzione di probabilità μ su $K \subset \mathbb{R}^n$ e ogni $N \geq 1$, esiste $f\_N \in \Sigma(\sigma, n)$ con N unità nascoste tale che*

$$\int\; |f(x)\; -\; fN\;(x)|2\;\; d\mu(x)\; \leq\; \frac{\left(2C\_f\right)}{N}$$

La velocità di convergenza O(N^{−1}) è cruciale: essa è **indipendente dalla dimensione n** dello spazio di input. Per i classici approssimatori lineari (polinomi, spline), la dipendenza è O(N^{−k/n}) per k derivate, un tasso che peggiora esponenzialmente con n (maledizione della dimensionalità). Le reti in Σ(σ,n) sfuggono a questa maledizione per la classe B_1.

La dimostrazione di Barron è probabilistica. Si parte dalla rappresentazione integrale di $f \in B_1$:

$$f(x)\;=\;f(0)\;+\;\int\; e\;^{i\omega^{\mathrm{T}}x}\,\hat{f}(\omega)\,d\omega$$

Si definisce la distribuzione di probabilità $p(\omega) = \|\omega\|\,|\hat{f}(\omega)| / C\_f$ su $\mathbb{R}^n$ e si campionano N frequenze $\omega_1, \ldots, \omega\_N$ i.i.d. da p. Per ogni campione ω_j si pone w_j = ω_j/‖ω_j‖ e b_j ~ Uniform[0, 2π]; il coefficiente di output c_j = C_f · (Re[f̂(ω_j) e^{iω^T_j·0}]/p(ω_j)) /N. Per la legge dei grandi numeri in spazio di Hilbert, la varianza dell'errore è al più $(2C^2\_f)/N$ per ogni N.

Poiché l'errore quadratico medio è la varianza, questo conclude la dimostrazione.

10.5 Separazione in profondità

> **Teorema 8.4 (Telgarsky, 2016).** *Per ogni r > 0 e ogni intero k ≥ 1, la funzione f_k costruita iterando k volte la funzione tent t(x) = max(0, min(x, 1−x, 2)) è calcolabile da una rete ReLU di profondità O(k) e larghezza O(1), ma richiederebbe larghezza Ω(2^{k/r}) per essere approssimata a errore minore di 1/r da una rete di profondità O(k/r).*

La costruzione di Telgarsky è esplicita: f_k(x) = t^{(k)}(x) (composizione k-fold della funzione tent t). Ogni applicazione di t raddoppia il numero di oscillazioni: t^{(k)} ha esattamente 2^{k-1} triangoli sull'intervallo [0,1]. Una rete shallow di larghezza W può produrre al più W triangoli; per approssimare f_k al livello ε occorre W = Ω(2^k), mentre la rete profonda usa O(k) neuroni. Il fattore esponenziale 2^k rappresenta il vantaggio composizionale della profondità.

Complementa questo risultato la formula di Montufar et al. (2014) sul numero massimo di regioni lineari create da una rete ReLU di profondità L con larghezza n per strato:

$$R(L,\ n) \leq \left(\sum_{j=0}^{n} \left(\frac{N}{j} \right) \right) \cdot \lfloor N/n \rfloor^{\ n(L-1)}$$

Per fissato numero di parametri totali, questo numero cresce esponenzialmente in L ma solo polinomialmente in N. Ogni strato aggiuntivo moltiplica il numero di regioni lineari per un fattore O(n), spiegando formalmente il vantaggio dell'architettura profonda.

10.6 Teoria statistica: dimensione VC e complessità di Rademacher

Dimensione VC. Per reti con attivazioni a soglia (Heaviside), Goldberg e Jerrum (1995) provarono VCdim(H_{W,L}) = Θ(WL log W). Bartlett, Harvey, Liaw e Mehrabian (2019) ottennero per reti ReLU:

$$VCdim(HW, L\ ) \ = \ O(WL\ log(W/L))$$

La dimensione VC controlla il comportamento di generalizzazione: con m esempi di addestramento, l'errore di generalizzazione è O(√(VCdim · log(m)/m)) con alta probabilità (teorema di Vapnik-Chervonenkis, 1971).

Complessità di Rademacher. Dato un training set S = {x_i}_{i=1}^m, la complessità di Rademacher empirica è

$$\hat{R}_S(H) \ = \ \mathbb{E}_\varepsilon\left[sup\_\{f\in H\}\ \frac{1}{m}\sum_{i=1}^{m}\ \varepsilon i\ \ f(xi\ )\right]$$

dove ε₁,…,ε_m sono variabili Rademacher i.i.d. in {−1,+1}. Il teorema di Rademacher-Bartlett-Mendelson (2002) afferma: con probabilità ≥ 1−δ su campioni i.i.d. da μ,

$$L(f) \leq \hat{L}S\ (f)\ +\ 2\hat{R}S\ (H)\ +\ 3\sqrt{\frac{log(2/\delta)}{2m}}$$

Bartlett, Foster e Telgarsky (2017) hanno ottenuto per reti neurali con pesi vincolati in norma spectral la stima

$$\hat{R}m\ (H) \leq \frac{c}{\sqrt{m}}\ (\sum_{\ell=1}^{L}\ \|W\ell\ \ ^{2/3}\ _F)\ ^{3/2}\sum_{\ell=1}^{L}\ \|W\ell\ \ ^{-2/3}\ _{2,\infty}$$

dove ‖·‖_F è la norma di Frobenius e ‖·‖_{2,∞} è la massima norma di riga. Questa stima è particolarmente utile per motivare le tecniche di normalizzazione spettrale nella regolarizzazione delle reti.

10.7 Il limite di larghezza infinita e la corrispondenza NNGP

Teorema 8.5 (NNGP, Lee et al. 2018). *Sia f_θ una rete feed-forward di profondità L con $w \sim N(0, \sigma^2_w/fan\_in)$, $b \sim N(0, \sigma^2_b)$. Il nucleo ricorsivo $K^{(\ell)}$ è definito da $K^{(0)}(x,x') = \sigma^2_w x^T x'/n + \sigma^2_b$ e $K^{(\ell+1)}(x,x') = \sigma^2_w \mathbb{E}_{(u,v)\sim N(0,\Lambda^{(\ell)})}[\sigma(u)\sigma(v)] + \sigma^2_b$, dove $\Lambda^{(\ell)} = [[K^{(\ell)}(x,x), K^{(\ell)}(x,x')],[K^{(\ell)}(x',x), K^{(\ell)}(x',x')]]$. Allora per $N\to\infty$, $f_\theta(\cdot) \to GP(0, K^{(L)})$ in distribuzione.*

La dimostrazione segue dal teorema centrale del limite funzionale. Per ogni strato ℓ, conditamente allo strato precedente, le preattivazioni $h^{(\ell)}_i(x) = \Sigma_j W^{(\ell)}_{ij} \varphi(h^{(\ell-1)}_j(x)) + b^{(\ell)}_i$ sono somme di N termini i.i.d. con varianza finita; per il TCL, converge a una gaussiana con media zero e covarianza $K^{(\ell+1)}$. L'indipendenza tra componenti diverse segue dall'indipendenza dei pesi.

Yang (2020) ha esteso questo risultato al framework Tensor Programs, mostrando che la propagazione del gradiente in reti di larghezza infinita è anch'essa descrivibile da processi gaussiani. Questo ha portato alla definizione del Neural Tangent Kernel (NTK, Jacot et al. 2018): nel limite di larghezza infinita, la dinamica di addestramento con gradient descent è equivalente alla regressione nel RKHS con nucleo

$$\Theta^{(L)}(x, x') = \sum_{\ell=0}^{L} \frac{\partial f}{\partial \theta}(x) \cdot \frac{\partial f}{\partial \theta}(x') \Big|_{\|\theta\|\to\infty}$$

Il NTK rimane costante durante l'addestramento nel limite di larghezza infinita, rendendo la dinamica lineare e quindi analiticamente trattabile. Questo è il motivo per cui le reti sufficientemente larghe mostrano comportamento di ottimizzazione convessa.

Capitolo 11 — Reinforcement Learning: MDP, Bellman, Policy Gradient e PPO

Il reinforcement learning studia come un agente autonomo impari a scegliere azioni in un ambiente dinamico al fine di massimizzare una misura cumulativa di ricompensa. A differenza dell'apprendimento supervisionato, nessun etichettatore esterno indica quale azione sia corretta; il segnale di apprendimento giunge unicamente dall'interazione con l'ambiente sotto forma di ricompense scalari ritardate e potenzialmente sparse.

11.1 Processi Decisionali di Markov

Un MDP è una quintupla $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$. Lo spazio degli stati $\mathcal{S}$ e quello delle azioni $\mathcal{A}$ possono essere continui (come lo spazio di configurazione di un robot); il nucleo di transizione $P : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$ soddisfa la proprietà di Markov; $R : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$ è la funzione di ricompensa; $\gamma \in [0,1)$ è il fattore di sconto. Il rendimento al tempo t è

$$G_t = \sum_{k=0}^{\infty} \gamma k \; Rt + k + 1$$

L'obiettivo è trovare π^* che massimizzi $J(\pi) = \mathbb{E}\_\pi[G_0]$. Una politica $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ è una mappa dagli stati alle distribuzioni sulle azioni.

11.2 Equazioni di Bellman

La funzione valore V^π e la Q-function Q^π soddisfano le equazioni di Bellman di attesa:

$$V^{\pi}(s) = \sum_{a} \pi(a|s)\left[r(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V^{\pi}(s')\right]$$

$$Q^{\pi}(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) \sum_{a'} \pi(a'|s')\, Q^{\pi}(s',a')$$

In forma matriciale V^π = r^π + γP^π V^π ammette soluzione unica V^π = (I − γP^π)^{−1} r^π poiché ‖γP^π‖_∞ = γ < 1. Le equazioni di ottimalità di Bellman:

$$V^{*}(s) = max_a \left[r(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V^{*}(s') \right]$$

$$Q^{*}(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a)\, max_{a'}\, Q^{*}(s',a')$$

L'operatore di Bellman ottimale T*V(s) = max_a [r(s,a)+γΣ_{s'}P(s'|s,a)V(s')] è una γ-contrazione su (ℝ^|𝒮|, ‖·‖_∞): ‖T*V − T*V'‖_∞ ≤ γ‖V−V'‖_∞. Per il teorema di Banach, V* è il punto fisso unico e la value iteration V_{k+1} = T*V_k converge geometricamente.

11.3 Programmazione Dinamica

Policy evaluation risolve iterativamente le equazioni di Bellman di attesa; converge con tasso γ per iterazione. Policy iteration alterna valutazione della politica corrente e miglioramento:

$$\pi'(s) \leftarrow arg\, max_a \left[r(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V^{\pi}(s') \right]$$

Il teorema di miglioramento della politica garantisce V^{π'}(s) ≥ V^π(s) ∀s. Poiché l'insieme delle politiche deterministiche è finito (|𝒜|^|𝒮|), policy iteration converge in un numero finito di passi.

11.4 TD Learning e Q-Learning

Con modello incognito, il TD error a un passo è

$$\delta_t = Rt + 1 + \gamma V(st + 1) - V(st)$$

Q-learning (Watkins, 1989):

$$Q(st\ ,at\ ) \leftarrow Q(st\ ,at\ )\ +\ \alpha\Big[Rt\ +\ 1\ \ +\ \gamma\ maxa'\ \ Q(st\ +\ 1\ ,a')\ -\ \ Q(st\ ,at\ )\Big]$$

Converge a Q* sotto le condizioni di Robbins-Monro ($\Sigma\alpha\_t = \infty$, $\Sigma\alpha^2\_t < \infty$) e visita completa degli stati-azioni (Watkins e Dayan, 1992).

11.5 Policy Gradient Theorem

> **Teorema 9.1 (Policy Gradient, Sutton et al. 1999).** *Per qualsiasi MDP e politica parametrica differenziabile π_θ: $\nabla\_\theta J(\theta) = \mathbb{E}\_{\pi\_\theta}[\nabla\_\theta \log \pi\_\theta(a|s) \cdot Q^{\pi\_\theta}(s,a)]$*

La dimostrazione sfrutta il log-derivative trick $\nabla\_\theta\ \pi\_\theta(a|s) = \pi\_\theta(a|s)\ \nabla\_\theta \log\ \pi\_\theta(a|s)$. Partendo dalla definizione $J(\theta) = \Sigma\_s\ d^{\pi\_\theta}(s)\ \Sigma\_a\ \pi\_\theta(a|s)\ Q^{\pi\_\theta}(s,a)$ dove d^{π_θ} è la distribuzione stazionaria degli stati:

$$\nabla\theta\ \ J(\theta)\ \ =\ \ \sum_{s}\ d\ \ ^{\pi\_\theta}(s) \sum_{a}\ Q\ \ ^{\pi}(s,a)\ \nabla\theta\ \ \pi\theta\ \ (a|s)$$

$$=\ \sum_{s}\ d\ \ ^{\pi\_\theta}(s) \sum_{a}\ \pi\theta\ \ (a|s)\ Q\ \ ^{\pi}(s,a)\ \nabla\theta\ \ log\ \pi\theta\ \ (a|s)$$

$$=\ \ \mathbb{E}\pi\_\theta\ \Big[Q\ \ ^{\pi}(s,a)\ \nabla\theta\ \ log\ \pi\theta\ \ (a|s)\Big]$$

Crucialmente, $\nabla\_\theta\ d^{\pi\_\theta}(s)$ non appare nel risultato perché la variazione della distribuzione stazionaria si cancella con la variazione del Q-value: questo è il senso più profondo del teorema. ∎

11.6 REINFORCE con Baseline e Generalized Advantage Estimation

REINFORCE stima il gradiente con Monte Carlo. Per ridurre la varianza si sottrae una baseline b(s). La scelta $b(s) = V^{\pi\_\varphi}(s)$ dà il fattore di

ponderazione A^π(s,a) = Q^π(s,a) − V^π(s). Il GAE (Schulman et al. 2016) con parametro $\lambda \in [0,1]$ interpola bias-varianza:

$$\hat{A}^{GAE(\gamma,\lambda)}t \;=\; \sum_{k=0}^{\infty} (\gamma\lambda)k \;\; \delta t + k \;\; con \;\; \delta t \;\; = \; Rt + 1 \;\; + \; \gamma V(st + 1\;) \; - \; V(st\;)$$

Per λ=0, Â^{GAE} = δ_t (TD residual, alta distorsione, bassa varianza). Per λ=1, Â^{GAE} = Σ_{k=0}^∞ γ^k δ_{t+k} = G_t − V(s_t) (stima Monte Carlo, bassa distorsione, alta varianza). La scelta $\lambda \approx 0.95$ bilancia entrambe le proprietà.

11.7 TRPO e la sua formulazione variazionale

TRPO (Schulman et al. 2015) massimizza il surrogate objective soggetto a un vincolo di trust region KL:

$$max\theta \quad \mathbb{E}s, a{\sim}\pi\_\{\theta\_old\} \left[\frac{\pi\theta \;(a|s)}{\pi\theta\_old \;(a|s)} \hat{A}(s,a) \right] \;\; s.t. \;\; \mathbb{E}s \left[KL(\pi\theta\_old \;\; \| \, \pi\theta \;)(s) \right] \leq \delta$$

Il vincolo KL viene risolto con il Natural Policy Gradient: il gradiente viene precondizionato con l'inversa della matrice di Fisher F = 𝔼[∇_θ log π_θ ∇_θ log π_θ^T]:

$$\Delta\theta \;=\; \alpha \, F^{\;-1} g, \quad g \;=\; \nabla\theta \;\; L\theta\_old \;\; (\theta) \;\; _{\theta=\theta\_old}$$

dove α è scelto con backtracking line search. Il natural gradient trasforma il gradiente euclideo in un gradiente nella metrica di Fisher, che è la metrica riemanniana naturale sullo spazio delle distribuzioni di probabilità. In questa metrica, il passo è invariante per riparametrizzazione di π_θ — una proprietà geometricamente desiderabile.

11.8 Proximal Policy Optimization (PPO)

PPO (Schulman et al. 2017) approssima il vincolo di TRPO con una penalizzazione soft attraverso la funzione obiettivo clippeata:

$$L^{CLIP}(\theta) = \mathbb{E}t\left[min\left(rt\ (\theta)\,\hat{A}t\ \right),\ clip\left(rt\ (\theta),\ 1-\varepsilon,\ 1+\varepsilon\right)\hat{A}t\ \right]$$

dove r_t(θ) = π_θ(a_t|s_t)/π_{θ_old}(a_t|s_t) e ε ≈ 0.2. Analisi per i due casi: per Â_t > 0, la funzione è min(r_t Â_t, (1+ε)Â_t), ovvero l'obiettivo è π_θ-lineare per r_t ≤ 1+ε e costante per r_t > 1+ε — massimizzare la probabilità dell'azione buona ma non eccessivamente. Per Â_t < 0, è min(r_t Â_t, (1−ε)Â_t), ovvero lineare per r_t ≥ 1−ε e costante per r_t < 1−ε — minimizzare la probabilità dell'azione cattiva ma non eccessivamente.

L'obiettivo completo con critic e entropia:

$$L^{PPO}(\theta) = \mathbb{E}t\left[L^{CLIP}t\ (\theta) - c_1(V\theta\ (st\ ) - Vtarg_t)^2 + c_2\,H[\pi\theta\ ](st\ )\right]$$

La variante con penalità KL adattiva aggiornaβ dopo ogni epoch: se KL > 1.5 d_targ, β ← 2β; se KL < d_targ/1.5, β ← β/2. Questo rende il vincolo effettivo autoregolante attorno al valore target d_targ.

11.9 Deep Q-Networks e Double DQN

DQN (Mnih et al. 2015) approssima Q* con una rete neurale $Q_\varphi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. L'addestramento minimizza l'errore di Bellman:

$$L_{DQN}(\varphi) = \mathbb{E}(s,a,r,s')\sim D\left[\left(r + \gamma\, maxa'\ Q\bar{\varphi}\ (s',a') - Q\varphi\ (s,a)\right)^2\right]$$

dove D è il replay buffer e $\bar{\varphi}$ sono i parametri della target network aggiornata con soft update $\bar{\varphi} \leftarrow \tau\varphi + (1-\tau)\bar{\varphi}$ ogni C passi. Double DQN (Van Hasselt et al. 2016) separa la selezione dell'azione dall'evaluation per ridurre il bias di sovra-stima:

$$target = r + \gamma\, Q\bar{\varphi}\ \left(s',\ arg\, maxa'\ Q\varphi\ (s',\ a')\right)$$

La separazione impedisce che la rete usi le sue stesse stime per valutare le azioni che ha scelto, eliminando il feedback loop che porta alla sovra-stima sistematica di Q.

11.10 Metodi Actor-Critic e Soft Actor-Critic (SAC)

I metodi actor-critic combinano l'approssimazione del valore (critic) con l'ottimizzazione diretta della policy (actor), superando i limiti di varianza di REINFORCE e i limiti di campionamento di Q-learning. Il critic stima il valore di azione $Q^\pi(s,a) = E[G_t \mid s_t=s, a_t=a]$, e l'actor usa questo valore per aggiornare la policy tramite il policy gradient. La versione soft del framework aggiunge un termine di entropia all'obiettivo per incoraggiare l'esplorazione:

$$J(\pi) = E_{\tau\sim\pi} [\sum_t r(s_t,a_t) + \alpha H(\pi(\cdot|s_t))] \quad (11.10)$$

dove $\alpha > 0$ è il coefficiente di temperatura e $H(\pi(\cdot|s)) = -E_{a\sim\pi}[\log \pi(a|s)]$ è l'entropia della policy nello stato s. Soft Actor-Critic (SAC, Haarnoja et al. 2018) massimizza questo obiettivo tramite un critic doppio per ridurre la sovrastima del valore (Double Q-learning) e un attore che parametrizza una distribuzione gaussiana squashed: $a = \tanh(\mu_\theta(s) + \sigma_\theta(s) \cdot \varepsilon)$, $\varepsilon \sim N(0,I)$. La temperatura α può essere ottimizzata automaticamente vincolando l'entropia attesa a un target $H_{target} = -\dim(A)$, tramite il problema duale lagrangiano: $\alpha^* = \arg\min_\alpha E[-\alpha \log \pi^*(a|s) - \alpha H_{target}]$. SAC è l'algoritmo di RL model-free più efficiente per spazi di azione continui ed è la baseline standard nei benchmark di manipolazione robotica (robosuite, sezione 30.4).

11.11 RL in Robotica: Reward Shaping, Curriculum e Trasferimento

L'applicazione del RL alla robotica presenta sfide specifiche che non emergono nei benchmark simulati standard: la rarefazione della ricompensa, la sicurezza durante l'esplorazione, e il trasferimento sim-to-real. La ricompensa sparsa ($r = 1$ solo al successo del task) rende l'esplorazione casuale inefficiente: per un task di inserimento peg-in-hole con un workspace di 1 m^3 e un foro di 5 mm di diametro, la probabilità

di successo di un policy casuale è O(10^{-8}), rendendo necessari miliardi di step di interazione. Il reward shaping definisce una funzione di ricompensa ausiliaria r_s(s,a) = r(s,a) + γΦ(s') - Φ(s), dove Φ(s) è un potenziale che guida l'esplorazione verso stati promettenti senza alterare la policy ottimale (invarianza del reward shaping, Ng et al. 1999).

Il curriculum learning adatta la difficoltà del task durante il training: inizia con task semplici (distanza iniziale piccola, tolleranze ampie) e aumenta progressivamente la difficoltà man mano che la policy migliora. Automatic Domain Randomization (ADR, Akkaya et al. 2019) implementa il curriculum a livello dei parametri del simulatore: il range di variazione dei parametri fisici viene ampliato gradualmente in base alla performance, raggiungendo policy che si trasferiscono a oggetti con massa, forma e attrito molto diversi da quelli di training. Il trasferimento sim-to-real per RL è trattato in dettaglio nel Capitolo 30; la connessione con i world model (Parte III) emerge nell'uso di world model addestrati in simulazione come ambienti di addestramento per la policy: DreamerV3 (sezione 17.7) usa il proprio world model interno come simulatore differenziabile, riducendo di 100x il numero di interazioni reali necessarie.

Capitolo 12 — Apprendimento per Imitazione: Behavioral Cloning, DAgger e GAIL

L'apprendimento per imitazione (Imitation Learning, IL) studia come un agente apprenda una policy a partire da dimostrazioni di un esperto, senza che sia necessario specificare una funzione di ricompensa esplicita. A differenza del reinforcement learning — che richiede di interagire con l'ambiente per raccogliere il segnale di ricompensa — l'IL sfrutta un dataset di traiettorie dimostrate, tipicamente raccolte da un operatore umano tramite teleoperazione o kinesthetic teaching. Questo paradigma è alla base dell'intera famiglia dei Vision-Language-Action models (RT-1, π_0, GR00T) e costituisce il meccanismo di fine-tuning dominante per la robotica manipolativa avanzata.

12.1 Behavioral Cloning: formulazione e limiti strutturali

Il caso più semplice dell'IL è il Behavioral Cloning (BC), che riduce il problema a regressione supervisionata. Dato il dataset di dimostrazioni $\mathcal{D}$ = {(o$_i$, a$_i$)}_{i=1}^N prodotto dalla policy esperta π_e, si minimizza la perdita di cross-entropia (per azioni discrete) o dei minimi quadrati (per azioni continue):

$$\mathcal{L}\_BC(\theta) \;=\; \mathbb{E}\_\{(o,a)\sim\mathcal{D}\}\,[\; -\,log\;\pi\_\theta(a \mid o)\;] \quad (9.1)$$

La semplicità computazionale del BC cela un limite strutturale fondamentale: il distributional shift. Durante l'addestramento, la policy π_θ ottimizza il proprio comportamento sulla distribuzione d^e degli stati visitati dall'esperto. Al deployment, tuttavia, π_θ incontra stati la cui distribuzione d^p si discosta da d^e, poiché i piccoli errori commessi dalla

policy si accumulano trascinando il sistema fuori dalla distribuzione di addestramento.

> **Teorema 9.1 (Ross–Gordon–Bagnell, 2011).** Sia T l'orizzonte temporale dell'episodio e $\varepsilon = \mathbb{E}_{o\sim d^e}[\mathscr{L}_{BC}(\pi_\theta, o)]$ la perdita media in-distribuzione. Allora il costo totale della policy appresa con BC soddisfa:
>
> $$J(\pi_\theta) \leq J(\pi_e) + \varepsilon T^2 \quad (9.2)$$
>
> Il bound è stretto: esiste una famiglia di MDP per cui $J(\pi_\theta) = \Omega(\varepsilon T^2)$.

Dimostrazione (schema).

Sia d_{π_θ} la distribuzione degli stati al passo t sotto la policy appresa π_θ. Per ogni passo t, la divergenza TV tra d_{π_θ} e d_e si accumula linearmente nell'errore ε per ogni step:

$$\|d_t^{\pi_\theta} - d_t^{\pi_e}\|_{TV} \leq t \cdot \varepsilon$$

poiché ogni step può aggiungere al più ε di divergenza cumulativa (per la definizione di ε come perdita media). Il costo aggiuntivo rispetto all'esperto è proporzionale alla deviazione $d_{\pi_\theta} - d_{\pi_e}$ pesata per T passi, dando il bound $T \cdot (T \cdot \varepsilon) = \varepsilon T^2$. La tight-ness si dimostra con un MDP a catena lineare dove ogni errore devia irreversibilmente dalla traiettoria ottima.

□

12.2 DAgger: Dataset Aggregation e riduzione al tasso lineare

L'algoritmo DAgger (Ross, Gordon e Bagnell 2011) riduce il bound quadratico (9.2) a uno lineare aggregando iterativamente nuove dimostrazioni raccolte negli stati effettivamente visitati dalla policy

corrente. L'esperto annota online le azioni corrette in questi stati, producendo coppie (o, a*) che vengono aggiunte al dataset.

> **Algoritmo 9.1 (DAgger).** Inizializzare $\mathcal{D}_1 = \varnothing$, π_1 esperto. Per iterazione i = 1, …, N: (1) raccogliere traiettorie con policy π_i, ottenendo stati {o◎}; (2) richiedere all'esperto le azioni a*◎ = π_e(o◎); (3) aggregare $\mathcal{D}$_{i+1} = $\mathcal{D}_i \cup \{(o◎, a*◎)\}$; (4) addestrare $\pi_{i+1} \in$ arg min_{π} $\mathcal{L}$_BC(π; $\mathcal{D}$_{i+1}).

> **Teorema 9.2 (Ross–Gordon–Bagnell, 2011).** Con N iterazioni di DAgger e ε_N = min_{π} $\mathbb{E}$_{o~d̄_N}[$\mathcal{A}$(π,o)] la perdita ottima al passo N sulla distribuzione miscelata d̄_N = (1/N)Σ d_i^{P}:

$$J(\pi\_N) \leq J(\pi_e) \;+\; \varepsilon\_N \cdot T \;+\; O(1/\sqrt{N}) \quad (9.3)$$

Il bound (9.3) mostra che DAgger riduce la dipendenza da T al primo ordine: l'errore cresce linearmente (non quadraticamente) nell'orizzonte T. Nella pratica della robotica, questo si traduce in policy che generalizzano significativamente meglio in presenza di perturbazioni che deviino il sistema dalla traiettoria nominale — esattamente il tipo di errore che affligge le dimostrazioni raccolte con kinesthetic teaching, dove l'operatore raramente porta il manipolatore in stati perturbati.

12.3 GAIL: Generative Adversarial Imitation Learning

GAIL (Ho e Ermon, 2016) interpreta l'IL come un problema di corrispondenza occupancy measure. L'occupancy measure di una policy π in un MDP è la distribuzione congiunta stato-azione ϱ_π(s,a) = π(a|s) Σ_{t=0}^∞ γ^t P(s_t = s | π). Due policy hanno comportamento identico (stesso ritorno atteso per ogni funzione di ricompensa) se e soltanto se ϱ_π_1 = ϱ_π_2 — risultato derivante dal duality theorem dell'ottimizzazione lineare del MDP (Syed e Schapire, 2007).

Anziché minimizzare $\mathcal{L}$_BC, GAIL minimizza la divergenza Jensen-Shannon tra ρ_{π_θ} e ρ_{π_e}:

$$\min_{\pi_\theta} \max_{D_w} \mathbb{E}_{\rho_{\pi_e}}[\log D_w(s,a)] + \mathbb{E}_{\rho_{\pi_\theta}}[\log(1 - D_w(s,a))] - \lambda H(\pi$$

(9.4)

dove $D_w : S \times A \to (0,1)$ è il discriminatore (rete neurale con parametri w) che distingue coppie (s,a) esperto da coppie generate dalla policy, e $H(\pi_\theta) = -\mathbb{E}[\log \pi_\theta(a|s)]$ è l'entropia della policy (agisce come regolarizzatore). L'obiettivo (9.4) è un gioco minimax: l'outer loop è RL (tipicamente TRPO o PPO) che minimizza sull'occupancy della policy usando $-\log D_w(s,a)$ come funzione di ricompensa surrogata; l'inner loop è un passo di aggiornamento del discriminatore.

All'equilibrio di Nash ($\pi_\theta = \pi_e$, $D_w = 1/2$ su tutto $S \times A$), l'obiettivo vale $-\log(4)$ e la divergenza JS è nulla. GAIL supera BC e DAgger nei compiti dove la funzione di ricompensa è difficile da specificare (e.g., locomotione in stile naturale) ma richiede interazione online con l'ambiente per raccogliere traiettorie da π_θ — un ostacolo pratico nella robotica fisica.

12.4 Action Chunking con Transformer: ACT

Zhao et al. (2023) introducono ACT (Action Chunking with Transformers), un'architettura di IL che affronta la sfida della compounding error mediante il concetto di action chunking: invece di predire una singola azione a_t, la policy genera un blocco di k azioni consecutive ($a_t, a_{t+1}, \ldots, a_{t+k-1}$) con un unico forward pass. Questo riduce di un fattore k il numero di query che il modello deve effettuare per un episodio, e di conseguenza il numero di punti in cui il distributional shift può manifestarsi.

L'architettura è un CVAE (Conditional Variational Auto-Encoder) con encoder-decoder Transformer. L'encoder riceve la sequenza di

osservazioni e il blocco di azioni esperto e produce un embedding latente $z \in \mathbb{R}^d$; il decoder, condizionato su z e sull'osservazione corrente, genera il blocco di azioni predetto. La perdita di addestramento è l'ELBO condizionale:

$$\mathcal{L}\_ACT = \mathbb{E}\_q[\|A\_\{t:t+k\} - \hat{A}\_\{t:t+k\}\|^2] + \beta\, KL[q\_\varphi(z|o, A) \,\|\, p(z)] \tag{9.5}$$

Il termine di ricostruzione minimizza l'errore quadratico medio sull'intero chunk, mentre il termine KL regolarizza il latente z verso una prior gaussiana N(0,I), con β che bilancia fedeltà e regolarizzazione. All'inferenza z è campionato dalla prior (non dall'encoder), rendendola un modello generativo puro condizionato sull'osservazione. ACT, addestrato su 50 dimostrazioni per task con un braccio bimanuale 14-DOF, raggiunge tasso di successo del 91% in task di manipolazione destra fine come il threading di viti, dimostrando che la struttura temporale del chunking compensa efficacemente il distributional shift.

12.5 Connessioni con i modelli VLA e la robotica moderna

L'apprendimento per imitazione è il meccanismo fondante dell'intera generazione dei Vision-Language-Action (VLA) models. RT-1 (Brohan et al. 2022) addestra un Transformer su 130.000 episodi di teleoperazione con BC puro, usando come input immagini RGB e istruzioni in linguaggio naturale e producendo azioni tokenizzate discretamente. π_0 (Black et al. 2024) estende il paradigma con una testa di flow matching su azioni continue, ottenendo controllo a 50 Hz. GR00T N1 (NVIDIA 2025) introduce la pre-training su video umani mediante LAPA (azioni latenti apprese senza etichette), poi fine-tuned con BC sulle dimostrazioni robot.

Il bound di DAgger (9.3) spiega quantitativamente perché il fine-tuning su dimostrazioni robot — anche poche decine — migliora radicalmente un modello pre-addestrato su larga scala: le dimostrazioni aggiornano la

distribuzione degli stati verso quella effettiva del deployment, riducendo ε_N. Il limite del BC puro (bound ε T²) è invece la motivazione principale del co-training sim-and-real (§13.5), che riduce ε includendo dati simulati perturbati dalla distribuzione di deployment.

Capitolo 13 — Meccanismi di Attenzione e Architettura Transformer

Il meccanismo di attenzione è la pietra angolare dell'architettura Transformer (Vaswani et al., 2017) e, per estensione, di tutti i modelli linguistici e visuomotori di grande scala che dominano la robotica avanzata contemporanea. Dal punto di vista matematico, l'attenzione realizza un'operazione di aggregazione ponderata su una sequenza di vettori, dove i pesi dipendono dal contenuto dei vettori stessi. Questo capitolo sviluppa la teoria completa: dalla scaled dot-product attention con motivazione statistica della scalatura, all'analisi dell'attenzione multi-testa, alla normalizzazione degli strati con bound espliciti, alle codifiche posizionali sinusoidali e RoPE.

13.1 Attenzione come problema di recupero morbido

Concettualmente, il meccanismo di attenzione risolve un problema di recupero morbido (soft retrieval). Si ha un insieme di coppie chiave-valore {(K_i, V_i)}_{i=1}^n e una query Q; l'obiettivo è restituire una combinazione lineare dei valori, dove i pesi riflettono la rilevanza di ogni chiave rispetto alla query.

Formalmente, dato uno spazio metrico su chiavi e query, l'attenzione hard associerebbe Q alla chiave più simile: output = V_{argmax_i sim(Q, K_i)}.

L'attenzione soft ammorbidisce questa operazione sostituendo il massimo con una somma pesata da una distribuzione di probabilità:

$$Attn(Q,\ K,\ V)\ =\ \sum_{i=1}^{n}\ \alpha i\ (Q,\ K)\cdot Vi$$

dove α_i(Q, K) = softmax_i(sim(Q, K_j)) per j = 1,…,n. La scelta della funzione di similarità determina il tipo di attenzione: il prodotto scalare è la scelta dominante per la sua efficienza computazionale e le sue proprietà geometriche.

13.2 Scaled Dot-Product Attention

Siano X ∈ ℝ^{n×d_model} la sequenza di n token con embedding di dimensione d_model. Si proiettano i token in spazi di query, chiave e valore attraverso matrici di proiezione W_Q ∈ ℝ^{d_model×d_k}, W_K ∈ ℝ^{d_model×d_k}, W_V ∈ ℝ^{d_model×d_v}:

$$Q\ =\ X\,W\_Q \in \mathbb{R}^{\{n\times d\_k\}},\quad K\ =\ X\,W\_K \in \mathbb{R}^{\{n\times d\_k\}},\quad V\ =\ X\,W\_V \in \mathbb{R}^{\{n\times d\_v\}}$$

La scaled dot-product attention è definita come

$$Attn(Q,\ K,\ V)\ =\ softmax\left(\frac{Q\,K^{T}}{\sqrt{dk}}\right)V$$

La matrice dei punteggi A = QK^T / √d_k ∈ ℝ^{n×n} ha l'elemento (i,j) che rappresenta la compatibilità tra il token i (come query) e il token j (come chiave). Dopo la softmax per righe, si ottiene la matrice dei pesi di attenzione α ∈ ℝ^{n×n} con α_{ij} ≥ 0 e Σ_j α_{ij} = 1. Il risultato finale è α V ∈ ℝ^{n×d_v}.

10.2.1 Motivazione statistica del fattore 1/√d_k

Il fattore di scala 1/√d_k non è arbitrario: ha una giustificazione statistica precisa. Consideriamo query e chiave come vettori casuali con

componenti i.i.d. con media zero e varianza unitaria: q_i ~ (0, 1) e k_j ~ (0, 1).

Il prodotto scalare q·k = Σ_{r=1}^{d_k} q_r k_r è una somma di d_k variabili aleatorie indipendenti. Per ogni r, q_r k_r ha:

$$\mathbb{E}[q\_r\, k\_r] = \mathbb{E}[q\_r]\cdot\mathbb{E}[k\_r] = 0, \quad Var[q\_r\, k\_r] = \mathbb{E}[q^2\_r]\cdot\mathbb{E}[k^2\_r] - (\mathbb{E}[q\_r]\cdot\mathbb{E}[k\_r])^2 = 1$$

Poiché le d_k coppie sono indipendenti, per la linearità della varianza:

$$Var[q\cdot k] = \sum_{r=1}^{d\_k} Var[q\_r\, k\_r] = d\_k$$

La deviazione standard del prodotto scalare grezzo è quindi √d_k. Senza scalatura, al crescere di d_k i logit A_{ij} = q_i · k_j hanno varianza crescente. Il softmax applicato a valori con varianza alta satura in regioni di gradiente quasi nullo: se un solo logit è molto più grande degli altri, il softmax assegna peso ≈ 1 a quel token e ≈ 0 a tutti gli altri, rendendo il gradiente ∂ softmax/∂A_{ij} ≈ 0 per quasi tutti i j. Formalmente, per la derivata del softmax:

$$\frac{\partial\, softmax(z)}{\partial z\_i}_{\,j} = softmax(z)\_j\,(\delta\_\{ij\} - softmax(z)\_i)$$

questa derivata tende a zero quando softmax(z)_j ≈ 0 o ≈ 1. La scalatura di A per 1/√d_k normalizza la varianza del prodotto scalare a 1 indipendentemente da d_k, preservando gradiente non degenerato lungo tutto l'addestramento. Più precisamente:

$$Var\left[\frac{q\cdot k}{\sqrt{dk}}\right] = \frac{Var[q\cdot k]}{d\_k} = \frac{d\_k}{d\_k} = 1$$

Un argomento più rigoroso considera la norma dei logit: per una query e d_k chiavi con norma O(1), la riga i della matrice dei punteggi ha norma O(√d_k) senza scalatura e O(1) con essa. Il soft-max a input con norma O(1) opera in un regime di gradiente stabile, dove la distribuzione di probabilità si distribuisce su più chiavi anziché concentrarsi su una sola.

13.3 Multi-Head Attention (MHA)

La Multi-Head Attention parallelizza h esecuzioni indipendenti di attenzione scaled dot-product con proiezioni distinte, consentendo al modello di catturare relazioni su sottospazi diversi dello spazio di rappresentazione.

Per la testa $\ell = 1,\ldots,h$, si definiscono le proiezioni

$W_{Q,\ell} \in \mathbb{R}^{d\_model \times d\_k}$, $W_{K,\ell} \in \mathbb{R}^{d\_model \times d\_k}$, $W_{V,\ell} \in \mathbb{R}^{d\_model \times d\_v}$

e la testa ℓ produce

$$head\ell \;=\; Attn(X\,WQ,\ell\;,\; X\,WK,\ell\;,\; X\,WV,\ell\;)$$

Le h teste vengono concatenate e proiettate nella dimensione originale tramite $W\_O \in \mathbb{R}^{hd\_v \times d\_model}$:

$$MHA(X) \;=\; Concat(head1\;,\;\ldots,\;headh\;)\,W\_O$$

La scelta tipica è d_k = d_v = d_model/h, cosicché ogni testa opera su un sottospazio di dimensione d_model/h e la concatenazione ripristina la dimensione originale. La complessità computazionale dell'MHA è $O(n^2 d\_model + n d^2\_model)$ per sequenza: il termine $O(n^2 d\_model)$ deriva dalla matrice di attenzione n×n, il termine $O(n d^2\_model)$ dalle proiezioni lineari.

Dal punto di vista espressivo, ogni testa può specializzarsi in un tipo distinto di relazione sintattica o semantica: alcune teste catturano dipendenze a breve distanza (attenzione locale), altre a lungo raggio (cross-sentence, cross-modal nel caso robotico). Questa specializzazione emerge spontaneamente dall'addestramento senza supervisione esplicita sul tipo di relazione.

10.3.1 Analisi della capacità espressiva dell'MHA

Cordonnier, Loukas e Jaggi (2020) hanno dimostrato che l'MHA con h teste e d_k dimensioni può simulare qualsiasi funzione booleana su n token: formalmente, per ogni matrice di permutazione P ∈ ℝ^{n×n} esiste una configurazione di pesi (W_Q, W_K, W_V, W_O) tale che MHA(X) = PX. Questo implica che l'MHA può implementare arbitrarie operazioni di copia, spostamento e mescolamento dei token.

Yun et al. (2020) hanno mostrato che un Transformer a 2 strati (con un singolo testa di attenzione per strato e una rete feed-forward) è un approssimatore universale di funzioni continue in norma L^2 su sequenze di lunghezza finita: per ogni $\varepsilon > 0$ e ogni $f \in C([0,1]^{n \cdot d})$, esiste un Transformer T_θ tale che $\|f - T_\theta\|_{L^2} \leq \varepsilon$. Il teorema si dimostra costruendo una sequenza hash che identifica ogni configurazione di input, poi usando le FFN per implementare la funzione desiderata su ogni hash.

13.4 Pre-Norm Residual e la stabilità del gradiente

I blocchi residui (He et al., 2016) sono fondamentali per addestrare reti molto profonde. In un Transformer, il blocco residuo è

$$y = x + f(x)$$

dove f può essere l'MHA o la rete feed-forward (FFN) a due strati. Il gradiente fluisce direttamente attraverso il percorso identità: $\partial y/\partial x = I + \partial f/\partial x$. Quando $\|\partial f/\partial x\|$ è piccolo (prima di convergenza), il gradiente è approssimativamente l'identità, evitando il vanishing gradient.

La versione originale del Transformer (Post-Norm) applica LayerNorm dopo l'operazione residua: y = LN(x + f(x)). La versione Pre-Norm (ora standard nei LLM) applica LayerNorm prima:

$$y = x + f(LN(x))$$

La Pre-Norm ha vantaggi di stabilità analiticamente quantificabili. Sia g = x + f(LN(x)); il gradiente rispetto a x è

$$\frac{\partial g}{\partial x} = I + \frac{\partial f(LN(x))}{\partial LN(x)} \cdot \frac{\partial LN(x)}{\partial x}$$

La derivata di LN è una proiezione ortogonale sulla ortocorsia (vedi §10.5), con norma spettrale ≤ 1. Questo limita il fattore Jacobiano del percorso attraverso la normalizzazione, impedendo esplosioni del gradiente. Xiong et al. (2020) hanno dimostrato formalmente che con Pre-Norm e learning rate sufficientemente piccolo, il gradiente dell'ultimo strato ha norma O(1/L) per una rete di profondità L, permettendo training stabile senza warm-up esteso.

13.5 Layer Normalization: derivazione e bound espliciti

La Layer Normalization (Ba et al., 2016) normalizza i d valori all'interno di ogni token indipendentemente dagli altri. Per un vettore x ∈ ℝ^d:

$$LN(x) = \gamma \odot \frac{x - \mu(x)\,1}{\sqrt{\sigma^2(x) + \varepsilon}} + \beta$$

dove μ(x) = (1/d) Σ_i x_i, σ²(x) = (1/d) Σ_i (x_i − μ)², γ,β ∈ ℝ^d sono parametri apprendibili (scala e traslazione), e ε > 0 è un termine di stabilità numerica (tipicamente 10^{−5}).

La derivata di LN rispetto a x è una matrice d×d calcolabile esplicitamente. Definiamo $\hat{x}$ = (x − μ1)/√(σ²+ε) il vettore normalizzato. Allora:

$$\frac{\partial LN}{\partial x}_{ij} = \frac{\gamma_i}{\sqrt{\sigma^2+\varepsilon}} \left(\delta_{ij} - \frac{1}{d} - \frac{\hat{x}_i \hat{x}_j}{d}\right)$$

dove δ_{ij} è il delta di Kronecker. In forma matriciale compatta:

$$\frac{\partial LN(x)}{\partial x} = \frac{1}{\sqrt{\sigma^2+\varepsilon}} Diag(\gamma) \cdot \left(I - \frac{11^T}{d} - \frac{\hat{x}\,\hat{x}^T}{d}\right)$$

Bound sulla norma spettrale di ∂LN/∂x. La matrice M = I − (11ᵀ)/d − ($\hat{x}$ $\hat{x}$ᵀ)/d è la proiezione ortogonale sul sottospazio di ℝ^d ortogonale a 1 e a $\hat{x}$. Poiché M è una proiezione, i suoi autovalori sono 0 o 1, quindi:

$$\|M\|_2 \leq 1, \quad \|\partial LN(x)/\partial x\|_2 \leq \|Diag(\gamma)\|_2 / \sqrt{(\sigma^2 + \varepsilon)} = \|\gamma\|_\infty / \sqrt{(\sigma^2 + \varepsilon)}$$

Per γ inizializzato a 1 e $\sigma^2 \geq 0$, il bound è $\|\gamma\|_\infty / \sqrt{\varepsilon} \approx 1/\sqrt{\varepsilon}$ per i passi iniziali. Nella pratica, con $\varepsilon = 10^{-5}$ si ha $\|\partial LN/\partial x\|_2 \leq 10^{2.5} \approx 316$ all'inizio; tuttavia $\sigma^2 > 0$ nella pratica riduce significativamente questo valore. L'analisi di Zhang et al. (2019) mostra che per sequenze tipiche $\sigma^2(x) \approx O(d\_model)$, cosicché il bound effettivo è $\|\gamma\|_\infty/\sqrt{(d\_model)}$, che è $O(1)$ per d_model grande.

La LayerNorm differisce dalla BatchNorm nell'asse di normalizzazione: BatchNorm normalizza attraverso il batch (per canale), LN normalizza attraverso i canali (per elemento del batch). Questa differenza è cruciale per i Transformer: la LN si comporta coerentemente per ogni lunghezza di sequenza e per batch size = 1, rendendo possibile l'inferenza single-example.

13.6 Posizioni e Codifiche Sinusoidali

Il meccanismo di attenzione è invariante per permutazione dei token: se si riordinano i token di input, l'output viene riordinato nello stesso modo (equivarianza alla permutazione). Questo è un pregio per task invarianti all'ordine, ma un difetto per modelli sequenziali dove l'ordine è semanticamente rilevante. Le codifiche posizionali (positional encodings, PE) iniettano informazione sull'ordine nel modello.

Vaswani et al. (2017) propongono codifiche posizionali sinusoidali. Per una sequenza di lunghezza n con embedding di dimensione d_model, la codifica della posizione pos nel canale (dimensione) 2i e 2i+1 è:

$$PE(pos, 2i) = sin\left(\frac{pos}{10000^{\frac{2i}{d\_model}}}\right), \quad PE(pos, 2i + 1) = cos\left(\frac{pos}{10000^{\frac{2i}{d\_model}}}\right)$$

Le frequenze angolari $\omega\_i = 1/10000^{2i/d\_model}$ formano una progressione geometrica da 1 (per i=0) a 1/10000 (per i=d_model/2−1),

coprendo scale di lunghezze d'onda da 2π a 10000·2π. L'embedding completo del token alla posizione pos è x_{pos} + PE(pos, ·).

Proprietà chiave delle PE sinusoidali. La motivazione è che la PE per una posizione relativa pos + k può essere espressa come trasformazione lineare della PE per pos:

$$PE(pos + k, 2i) = PE(pos, 2i)\, cos(\omega_i\, k) + PE(pos, 2i + 1)\, sin(\omega_i\, k)$$

$$PE(pos + k, 2i + 1) = -PE(pos, 2i)\, sin(\omega_i\, k) + PE(pos, 2i + 1)\, cos(\omega_i\, k)$$

In forma matriciale, la trasformazione è una rotazione di angolo ω_i k nel piano (2i, 2i+1). Questo implica che il prodotto scalare PE(pos)·PE(pos') = Σ_i cos(ω_i(pos−pos')) dipende solo dalla distanza relativa |pos − pos'| e non dalle posizioni assolute, una proprietà desiderabile per la generalizzazione a sequenze di lunghezza diversa da quella di addestramento.

13.7 Rotary Position Embedding (RoPE)

Le PE sinusoidali additive modificano il vettore di embedding prima dell'attenzione; RoPE (Su et al., 2021) integra l'informazione posizionale direttamente nella computazione del prodotto scalare tra query e chiave, con vantaggi di generalizzazione e di compatibilità con l'attenzione causale.

L'idea di RoPE è applicare una rotazione dipendente dalla posizione ai vettori q e k prima del prodotto scalare. Per d_k = 2 (caso base), la rotazione di posizione m è la matrice

$$R_m = [cos(m\theta) \quad -sin(m\theta) sin(m\theta) \quad cos(m\theta)]$$

Per d_k dimensioni generali, RoPE applica rotazioni bidimensionali indipendenti su coppie di componenti (2i, 2i+1) per i = 0,…,d_k/2−1, con angolo di base θ_i = 10000^{−2i/d_k}:

$$R^{(m)}_{d\_k} = Diag(Rm\theta\_0\ ,\ Rm\theta\_1\ ,\ \ldots,\ Rm\theta\_\{d\_k/2 - 1\}\ )$$

dove il blocco diagonale i-esimo è la matrice di rotazione 2×2 di angolo mθ_i. Il prodotto scalare chiave-query con RoPE diventa:

$$qm^{\mathrm{T}} kn = (R^{(m)} q)^{\mathrm{T}} (R^{(n)} k) = q^{\mathrm{T}} R^{(n-m)} k$$

dove si è usata la proprietà R^{(m)ᵀ} R^{(n)} = R^{(n-m)} per matrici di rotazione. Il prodotto scalare dipende solo dalla posizione relativa n−m, come nelle PE sinusoidali, ma la codifica è integrata nella trasformazione di q e k anziché essere aggiunta all'embedding.

In forma equivalente, per un vettore x ∈ ℝ^{d_k} scritto come sequenza di blocchi di dimensione 2, x = [(x_0, x_1), (x_2, x_3), …], la rotazione RoPE è:

$$R^{(m)} x = [x\_0\ cos(m\theta\_0) - x\_1\ sin(m\theta\_0) x\_0\ sin(m\theta\_0) + x\_1\ cos(m\theta\_0) x\_2\ cos(m\theta\_1)$$

che si può calcolare efficientemente come prodotto element-wise senza costruire esplicitamente la matrice R:

$$R^{(m)} x = x \odot cos(m\Theta) + xrot \odot sin(m\Theta)$$

dove x^{rot} è x con le coppie ruotate di 90° (ovvero x^{rot} = [(−x_1, x_0), (−x_3, x_2), …]) e Θ = (θ_0, θ_0, θ_1, θ_1, …). Questo riduce la complessità da O(d²_k) a O(d_k) per posizione.

RoPE ha un'ulteriore proprietà di decadimento: il prodotto scalare ‖q‖ ‖k‖ Σ_i cos((m−n)θ_i)/d_k decresce con la distanza |m−n| in modo approssimativamente monotono, fornendo un bias induttivo verso relazioni locali che si è mostrato empiricamente utile per la comprensione del linguaggio e per il controllo robotico sequenziale.

13.8 Feed-Forward Network nel Transformer

Ogni strato Transformer include, dopo il blocco MHA, una rete feed-forward a due strati (FFN) applicata in modo identico e indipendente a ogni posizione:

$$FFN(x) = W\_2 \cdot max(0, W\_1\, x + b\_1) + b\_2$$

dove W_1 ∈ ℝ^{d_{ff}×d_{model}}, W_2 ∈ ℝ^{d_{model}×d_{ff}} con d_{ff} = 4 d_{model} (tipicamente). In termini di ruoli, l'MHA implementa la comunicazione tra token (mixing di posizioni), mentre la FFN implementa l'elaborazione contestuale locale (trasformazione per-token). La separazione di questi due ruoli è analiticamente motivata e consente una specializzazione funzionale durante l'addestramento.

Varianti recenti usano attivazioni GeLU o SiLU anziché ReLU, e sostituiscono la FFN a due strati con la Gated Linear Unit (Dauphin et al. 2017):

$$GLU(x) = (W\_1\, x + b\_1) \odot \sigma(W\_3\, x + b\_3), \quad FFN\_\{GLU\}(x) = W\_2 \cdot GLU(x)$$

che introduce un controllo dinamico del flusso di informazione attraverso il meccanismo di gating. Le varianti SwiGLU e GeGLU (Noam Shazeer, 2020) usano rispettivamente Swish e GeLU come funzione di gating e sono adottate in LLaMA, PaLM e altri LLM moderni.

13.9 Complessità computazionale e attenzione efficiente

La complessità dell'attenzione standard è $O(n^2 d_k)$ in tempo e $O(n^2)$ in memoria (per la matrice di attenzione). Per sequenze lunghe — immagini ad alta risoluzione, traiettorie robotiche a lunga durata — questo diventa proibitivo. Sono stati proposti numerosi meccanismi di attenzione efficiente:

Attenzione lineare (Katharopoulos et al. 2020). Si sostituisce $\text{softmax}(QK^T/\sqrt{d_k})\, V$ con una decomposizione kernel: $\text{Attn}(Q,K,V) \approx \varphi(Q)(\varphi(K)^T V)$ dove $\varphi : \mathbb{R}^{d_k} \to \mathbb{R}^r$ è una mappa di features con $r \ll n$.

Sfruttando l'associatività della moltiplicazione matriciale, si calcola prima $\varphi(K)^T V \in \mathbb{R}^{r \times d_v}$ e poi $\varphi(Q)$ per questa matrice, riducendo la complessità a $O(nrd)$.

FlashAttention (Dao et al. 2022). Non riduce la complessità asintotica $O(n^2)$ ma riorganizza il calcolo in blocchi tile-based per sfruttare la gerarchia di memoria GPU (SRAM vs HBM), riducendo il numero di accessi alla memoria e ottenendo un'accelerazione pratica di 2-4× senza perdita di qualità.

Nel contesto di $\pi 0$ (Black et al. 2024), la sequenza di input al Transformer include token di immagine (prodotti dall'encoder SigLIP di PaliGemma), token di linguaggio e token di azione. La lunghezza totale è $O(n_{img} + n_{lang} + H)$ dove $H = 50$ è l'action chunk; con n_{img} tipicamente nell'ordine di centinaia di token per immagine, la gestione efficiente della matrice di attenzione è fondamentale per il controllo a 50 Hz.

13.10 Analisi della profondità e della larghezza nei Transformer

I Transformer presentano fenomeni di scalatura (scaling laws, Kaplan et al. 2020) che non si osservano nelle CNN: la loss su un corpus di linguaggio decresce come una legge di potenza nel numero di parametri N, nel volume di dati D e nel compute C:

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

con esponenti $\alpha_N \approx 0.076$, $\alpha_D \approx 0.095$, $\alpha_C \approx 0.050$ (stima per modelli di linguaggio puri). Queste leggi predicono la prestazione ottimale dato un budget computazionale fisso, guidando le decisioni architetturali per modelli come PaLM-E (540B parametri) e RT-2 (55B parametri).

Hoffmann et al. (2022) — Chinchilla — hanno mostrato che i modelli precedenti erano overparametrizzati rispetto ai dati: la legge di Chinchilla

afferma che il numero ottimo di token di addestramento D* è circa 20N per un modello di N parametri. Questo ha ridirizzato la ricerca verso modelli più piccoli addestrati su dataset più grandi, con implicazioni dirette per i robot foundation models dove i dati sono costosi da raccogliere.

Capitolo 14 — Modelli Generativi: VAE, Diffusione e Flussi Normalizzanti

I modelli generativi apprendono una distribuzione implicita o esplicita sui dati per campionare nuove istanze, interpolare tra configurazioni note, o stimare densità. Nel contesto robotico trovano applicazione nella rappresentazione di distribuzioni multimodali di politiche, nella sintesi di traiettorie, nella generazione di sequenze di azioni denoising e nella modellazione di distribuzioni su spazi di azioni ad alta dimensione.

14.1 Modelli Latenti e ELBO

Un modello generativo latente postula variabili latenti $z \in \mathbb{R}^d$ che spiegano $x \in \mathbb{R}^n$ attraverso $p_\theta(x|z)$ con prior $p(z)$. La distribuzione marginale $p_\theta(x) = \int p_\theta(x|z)p(z)dz$ è tipicamente intrattabile. L'ELBO si deriva dall'identità

$$log\, p\theta\ (x)\ =\ \mathcal{L}(\theta, \varphi; x)\ +\ KL(q\varphi\ (z|x) \,\|\, p\theta\ (z|x))$$

con $\mathcal{L}(\theta,\varphi;x) = \mathbb{E}_{q_\varphi(z|x)}[\log p_\theta(x|z)] - KL(q_\varphi(z|x) \,\|\, p(z))$. Poiché $KL \geq 0$, $\mathcal{L} \leq \log p_\theta(x)$. Massimizzare $\mathcal{L}$ rispetto a θ, φ: (i) massimizza la verosimiglianza di ricostruzione; (ii) minimizza la divergenza encoder-prior (regolarizzazione).

14.2 Variational Autoencoder

Il VAE (Kingma e Welling, 2014) implementa l'ELBO con encoder $q_\varphi(z|x) = N(z; \mu_\varphi(x), diag(\sigma^2_\varphi(x)))$ e decoder $p_\theta(x|z)$. Il reparametrization trick consente la backpropagation: $z = \mu_\varphi(x) + \sigma_\varphi(x) \odot \varepsilon$, $\varepsilon \sim N(0,I)$.

La KL closed-form tra gaussiane diagonali è:

$$KL(q\varphi\ (z|x) \parallel N(0,I)) \ = \ \frac{1}{2}\sum_{j=1}^{d}\left(\mu 2^{j} \ + \ \sigma 2^{j} \ - \ log\ \sigma 2^{j} \ - \ 1\right)$$

La perdita totale su M esempi:

$$\mathcal{L}(\theta,\varphi) \ = \ \frac{1}{M}\sum_{i=1}^{M}\left[\mathbb{E}\varepsilon\left[log\ p\theta\ (x(i)\ |z(i)\ )\right] \ - \ KL(q\varphi\ (z|x(i)\ ) \parallel p(z))\right]$$

14.3 Modelli di Diffusione DDPM

11.3.1 Il processo forward

Il processo forward aggiunge rumore gaussiano per T passi:

$$q(xt\ |xt-1\ ) \ = \ N(xt\ ;\ \sqrt{1-\beta t}\ xt-1\ ,\ \beta t\ \ I)$$

con $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$, la marginale chiusa è:

$$q(xt\ |x0\ ) \ = \ N(xt\ ;\ \sqrt{\bar{\alpha}t}\ x_0,\ (1-\bar{\alpha}t\ )\ I)$$

La derivazione usa la composizione di gaussiane: $x_t = \sqrt{(\alpha_t)}x_{t-1} + \sqrt{(\beta_t)}\varepsilon_t$ con $\varepsilon_t \sim N(0,I)$, e per sostituzione ricorsiva si ottiene $x_t = \sqrt{\bar{\alpha}_t}\ x_0 + \sqrt{(1-\bar{\alpha}_t)}\ \varepsilon$ con $\varepsilon \sim N(0,I)$. La varianza si conserva: $\text{var}(\sqrt{\alpha_t}\ x_{t-1}) + \text{var}(\sqrt{\beta_t}\ \varepsilon_t) = \alpha_t(1-\bar{\alpha}_{t-1}) + \beta_t = (1-\bar{\alpha}_t)$.

11.3.2 Il processo reverse e la perdita DDPM

La reverse condizionale $q(x_{t-1}|x_t, x_0)$ è gaussiana con

$$\tilde{\mu}t\ (xt\ ,x_0) \ = \ \frac{\sqrt{\bar{\alpha}t-1}\ \beta t}{1-\bar{\alpha}t}\ x_0 \ + \ \frac{\sqrt{\alpha t}\ (1-\bar{\alpha}t-1\ )}{1-\bar{\alpha}t}\ xt$$

derivata da Bayes: $q(x_{t-1}|x_t, x_0) \propto q(x_t|x_{t-1})\ q(x_{t-1}|x_0)$. Esprimendo μ_θ come predittore del rumore $\hat{\varepsilon}_\theta(x_t, t)$:

$$\mu\theta\ (xt\ ,t) \ = \ \frac{1}{\sqrt{\alpha t}}\left(xt \ - \ \frac{\beta t}{\sqrt{1-\bar{\alpha}t}}\ \hat{\varepsilon}\theta\ (xt\ ,t)\right)$$

la perdita semplificata di Ho et al. (2020) è:

$$L_{simple} = \mathbb{E}t, x_0, \varepsilon \left[\| \varepsilon - \hat{\varepsilon}\theta \left(\sqrt{\bar{\alpha}t}\ x_0 + \sqrt{1 - \bar{\alpha}t}\ \varepsilon,\ t \right) \|^2 \right]$$

11.3.3 DDIM e la prospettiva SDE

Il processo DDIM (Song et al. 2020) definisce la regola di aggiornamento deterministica (η=0):

$$xt - 1 = \sqrt{\bar{\alpha}t - 1} \left(\frac{xt - \sqrt{1-\bar{\alpha}t}\ \hat{\varepsilon}\theta\ (xt, t)}{\sqrt{\bar{\alpha}t}} \right) + \sqrt{1 - \bar{\alpha}t - 1}\ \hat{\varepsilon}\theta\ (xt, t)$$

La formulazione SDE continua è: dx = f(x,t)dt + g(t)dW, con processo reverse dx = [f(x,t) − g(t)² ∇_x log p_t(x)]dt + g(t)dW̄. La rete approssima lo score: ε̂_θ(x_t,t) ≈ −√(1−ᾱ_t) ∇_{x_t} log q_t(x_t).

11.3.4 Diffusion Policy nell'apprendimento visuomotorio

Chi et al. (2023) rappresentano π_θ(A_t|O_t) come DDPM condizionale. L'EBM implicito è p_θ(a|o) = e^{−E_θ(a,o)}/Z(o,θ), il cui score è

$$\nabla a\ log\ p(a|o) = -\nabla a\ E\theta\ (a, o) - \nabla a\ log\ Z(o, \theta) = 0 \approx -\varepsilon\theta\ (a, o)$$

L'assenza di Z nel gradiente garantisce la stabilità dell'addestramento rispetto agli EBM classici che richiedono negative sampling.

14.4 Flussi Normalizzanti

Un flusso normalizzante apprende una biiezione f_θ : ℝ^n → ℝ^n con log-verosimiglianza esatta:

$$log\ pX\ (x) = log\ pZ\ (f\theta\ (x)) + log\ |det\ Jf\_\theta\ (x)|$$

I coupling flows (RealNVP) partizionano x = (x_{1:d}, x_{d+1:n}) e applicano z_{d+1:n} = x_{d+1:n} ⊙ exp(s(x_{1:d})) + t(x_{1:d}) con det J = exp(Σ_i s_i(x_{1:d})) calcolabile in O(n). I CNF (Chen et al. 2018)

sostituiscono le trasformazioni discrete con dz/dt = v_θ(z,t) e la log-densità evolve secondo

$$\frac{d \log p(z(t))}{dt} = -tr\left(\frac{\partial v\theta}{\partial z}(z(t), t)\right)$$

La traccia viene stimata con il trucco di Hutchinson: tr(J) ≈ $\varepsilon^T J\varepsilon$ con ε ~ N(0,I), riducendo il costo da $O(n^2)$ a O(n).

14.5 Flow Matching

Il flow matching (Lipman et al. 2022; Liu et al. 2022) addestra v_θ a seguire percorsi di trasporto ottimale. Per il percorso lineare $x^\tau = \tau x_1 + (1-\tau)\varepsilon$ con ε ~ N(0,I), il campo vettoriale condizionale è $u(x^\tau | x_1) = x_1 - \varepsilon = x_1 - x^0$, e la perdita è:

$$L_{CFM}(\theta) = \mathbb{E}\tau, x_1, \varepsilon \left[\| v\theta\ (x^{\ \tau}, \tau) - (x_1 - \varepsilon)\|^{\ 2}\right]$$

In π0, l'action expert viene addestrato con questa perdita condizionale su chunk di azioni A_t. All'inferenza, le azioni si generano con Euler forward: $A^{\tau+\delta}_t = A^\tau_t + \delta\, v_\theta(A^\tau_t, o_t)$, partendo da $A^0_t \sim N(0,I)$ con 10 passi (δ=0.1). La corrispondenza tra flow matching e diffusione è che entrambi apprendono un trasporto da prior a dati, ma il flow matching usa percorsi lineari con stima diretta del campo vettoriale anziché predizione del rumore.

14.6 Connessioni e Prospettive Unificate

VAE, diffusione e flussi condividono la struttura di trasporto tra una prior gaussiana p_0 e la distribuzione dei dati p_data, differenziandosi nel metodo: il VAE usa un trasporto stocastico approssimato con gap ELBO; i flussi un trasporto deterministico esatto con vincolo di invertibilità; la diffusione un trasporto graduale su T passi con log-verosimiglianza lower bound.

Song et al. (2021) unificano diffusione e flussi: il processo SDE con rumore η interpola continuamente tra DDPM ($\eta=\sqrt{\beta\_t}$) e ODE deterministica ($\eta=0$). Albergo et al. (2023) generalizzano al Stochastic Interpolant framework: $x^{\tau} = \alpha(\tau)x_0 + \beta(\tau)x_1 + \gamma(\tau)\varepsilon$ è una famiglia one-parameter che include DDPM ($\beta=1$, $\alpha=\sqrt{(1-\tau)}$, $\gamma=\sqrt{\tau}$) e flow matching ($\beta=\tau$, $\alpha=1-\tau$, $\gamma=0$) come casi particolari.

Per i sistemi robotici avanzati, la scelta del paradigma generativo ha implicazioni dirette: RT-2 e PaLM-E usano autoregressionecon discretizzazione delle azioni (bassa frequenza, buona semantica); Diffusion Policy usa DDPM per multimodalità e stabilità; $\pi0$ usa flow matching per alta frequenza (50 Hz) e precisione. La tendenza attuale è verso architetture ibride che combinano il backbone linguistico-visivo per la semantica con un modulo di generazione continua (diffusione o flow matching) per il controllo motorio.

14.7 Modelli Generativi per la Robotica: Diffusion Policy e World Models Visivi

I modelli generativi descritti in questo capitolo trovano applicazione diretta nella robotica incarnata in due ruoli distinti: come policy generative e come world model percettivi. Nel ruolo di policy, i modelli di diffusione generano distribuzioni multimodali sulle azioni condizionate sull'osservazione: Diffusion Policy (Chi et al. 2023) usa un modello DDPM per apprendere $\pi(a \mid o)$ come processo di de-noising, producendo policy che gestiscono naturalmente le ambiguità nei dati di dimostrazione (due dimostrazioni per lo stesso stato possono scegliere azioni diverse, e il modello di diffusione le apprende tutte). La qualità della policy aumenta con il numero di passi di de-noising K, ma a costo di latenza: K=100 passi a 10 ms ognuno introducono un ritardo di 1 s, incompatibile con il controllo in tempo reale. Le varianti accelerate (DDIM, Consistency Models, Flow Matching) riducono K a 1-10 passi mantenendo la qualità.

$$a_{k-1} = \alpha_k a_k - \beta_k \varepsilon_\theta(a_k, k, o) + \sigma_k z, \quad z \sim N(0,I) \quad (14.7)$$

Nel ruolo di world model percettivo, i modelli generativi visivi (VideoGPT, MCVD, DreamBooth-Video) generano previsioni di frame futuri condizionate sulle azioni: $o_{t+1} \sim p_\theta(o_{t+1} \mid o_{1:t}, a_t)$. Questa capacità di predizione visiva è utile per la pianificazione visiva (UniPi, sezione 30.2) e per la generazione di dati sintetici di addestramento. Il prezzo è la complessità di modellare tutto lo spazio pixel, inclusi texture, illuminazione e rumore non rilevanti per il controllo. La transizione verso world model predittivi nel latente (RSSM, Dreamer, JEPA) è motivata da questa inefficienza: predire nello spazio latente piuttosto che nello spazio pixel riduce la complessità del modello di un ordine di grandezza, accelerando il training e migliorando la qualità della rappresentazione per il controllo. Questa transizione è il tema centrale delle Parti III e IV della monografia.

14.8 VAE nella Robotica: Rappresentazioni Latenti per la Manipolazione

Il Variational Autoencoder (sezione 14.2) è ampiamente usato nella robotica per comprimere osservazioni ad alta dimensione (immagini RGB-D, nuvole di punti) in rappresentazioni latenti a bassa dimensione adatte al controllo. In questo contesto, il decoder di ricostruzione non è il componente critico: ciò che importa è la qualità dello spazio latente Z come rappresentazione compatta dello stato del robot e dell'ambiente. Latent Space Roadmaps (Casas et al. 2020) usano un VAE per costruire una PRM (sezione 27.3) nello spazio latente: i nodi della roadmap sono configurazioni latenti $z_i = \mu_\varphi(o_i)$, e gli archi connettono configurazioni raggiungibili con un interpolatore latente lineare. Il planning dal latente z_start al latente z_goal si riduce a una ricerca sul grafo, con esecuzione via decodifica sequenziale delle configurazioni intermedie.

Un risultato di rilievo per la robotica è che la struttura dello spazio latente VAE può riflettere la struttura geometrica del workspace robotico: se il VAE è addestrato su immagini di traiettorie fisicamente plausibili, la metrica latente $\|z_1 - z_2\|$ approssima la distanza geodetica in C_free (sezione 27.1). Questa proprietà, sfruttata da Rappresentazioni Latenti per la Pianificazione del Moto (Ichter e Pavone 2019), permette di usare la distanza latente come euristica di costo per RRT* nel latente, combinando i vantaggi della completezza probabilistica di RRT* con la compattezza della rappresentazione latente. La connessione con JEPA (Parte IV) è diretta: JEPA rimpiazza l'obiettivo di ricostruzione del VAE con un obiettivo di predizione nel latente, ottenendo rappresentazioni più strutturate e più adatte al controllo.

Capitolo 15 — Teoria dell'Ottimizzazione per il Deep Learning

L'addestramento di una rete neurale profonda è, in termini formali, un problema di ottimizzazione non convessa in uno spazio di parametri ad altissima dimensione. La comprensione matematica del paesaggio di perdita — la geometria della funzione obiettivo nello spazio dei pesi — è fondamentale per spiegare il successo empirico del gradient descent stocastico e dei suoi raffinamenti adattativi. Il presente capitolo tratta la geometria della superficie di perdita, la convergenza di SGD con analisi quantitativa per funzioni L-lisce e μ-fortemente convesse, la derivazione completa dell'ottimizzatore Adam con le correzioni di bias, e l'analisi teorica della Batch Normalization come regolarizzatore del paesaggio di perdita.

15.1 Geometria della superficie di perdita

Sia $L : \mathbb{R}^d \to \mathbb{R}$ la funzione di perdita di una rete con d parametri. I punti critici di L sono i punti $\theta^* \in \mathbb{R}^d$ per cui $\nabla L(\theta^*) = 0$. Questi si suddividono in minimi locali, massimi locali e punti di sella (saddle points), classificati dallo spettro della matrice hessiana $H(\theta^*) = \nabla^2 L(\theta^*)$.

Classificazione dei punti critici. Un punto critico θ^* è:

un *minimo locale* se tutti gli autovalori di $H(\theta^*)$ sono strettamente positivi ($H > 0$);

un *massimo locale* se tutti gli autovalori di $H(\theta^*)$ sono strettamente negativi ($H < 0$);

un *punto di sella* se $H(\theta^*)$ ha autovalori sia positivi che negativi;

un *punto di sella degenere* se H(θ*) ha autovalori nulli con segno indefinito sul kernel.

Per reti sufficientemente larghe e dati realistici, Bhojanapalli et al. (2016), Du e Lee (2018) e altri hanno dimostrato che i minimi locali di reti overparametrizzate tendono ad essere globali o quasi-globali: la perdita in tutti i minimi locali è approssimativamente la stessa. Formalmente, per reti neurali con larghezza $N \to \infty$ addestrate con un dataset di m esempi, tutti i minimi locali sono ε-vicini al minimo globale con $\varepsilon \to 0$ al crescere di N/m.

La sfida principale non è l'esistenza di minimi locali cattivi, ma quella dei punti di sella. Dauphin et al. (2014) hanno argomentato che i punti critici in alta dimensione tendono ad essere selle: per funzioni con elementi hessiani random (ipotesi semicircle law di Wigner), la probabilità che tutti gli autovalori abbiano lo stesso segno decresce esponenzialmente con d. Tuttavia, i punti di sella del primo ordine (saddle points stricto sensu) non intrappolano il gradient descent: il gradiente non è zero nelle vicinanze e SGD sfugge naturalmente. I punti di sella degeneri (flat saddles) possono rallentare la convergenza.

15.2 Analisi di convergenza dell'SGD

Il gradient descent deterministico (GD) aggiorna i parametri secondo

$$\theta t + 1 \quad = \quad \theta t \quad - \quad \eta \, \nabla L(\theta t \,)$$

dove $\eta > 0$ è il learning rate (passo). Il comportamento di convergenza dipende crucialmente dalle proprietà di L.

Definizione 12.1 (L-smoothness). *Una funzione differenziabile $L : \mathbb{R}^d \to \mathbb{R}$ è L-smooth (o L-Lipschitz nel gradiente) se $\|\nabla L(\theta) - \nabla L(\theta')\| \leq L \|\theta - \theta'\|$ per ogni $\theta, \theta' \in \mathbb{R}^d$.*

La L-smoothness equivale a: la hessiana è limitata $\|H(\theta)\|_2 \le L$ per ogni θ. In termini pratici, L è la curvatura massima della funzione di perdita.

Definizione 12.2 (μ-convessità forte). *Una funzione differenziabile L è μ-fortemente convessa se* $L(\theta') \ge L(\theta) + \nabla L(\theta)^T(\theta'-\theta) + (\mu/2)\|\theta'-\theta\|^2$ *per ogni* $\theta, \theta' \in \mathbb{R}^d$. *Equivalentemente,* $H(\theta) \succcurlyeq \mu I$ *per ogni* θ.

12.2.1 Convergenza di GD per funzioni L-smooth

Teorema 12.1 (Convergenza GD per funzioni L-smooth). *Sia* $L : \mathbb{R}^d \to \mathbb{R}$ *funzione L-smooth con minimo globale* L^*. *Con learning rate* $\eta \le 1/L$, *il gradient descent soddisfa*

$$L(\theta t\ ) - L^* \le \frac{\|\theta_0 - \theta^*\|^2}{2\eta t}$$

La dimostrazione si ottiene sfruttando la disuguaglianza di discesa garantita dalla L-smoothness. Per ogni θ e passo $\eta \le 1/L$, vale

$$L(\theta - \eta\nabla L(\theta)) \le L(\theta) - \eta\|\nabla L(\theta)\|^2 + \frac{L\eta^2}{2}\|\nabla L(\theta)\|^2 = L(\theta) - \eta\left(1 - \frac{L\eta}{2}\right)\|\nabla L(\theta)\|^2$$

Per $\eta \le 1/L$, il fattore $(1 - L\eta/2) \ge 1/2$, quindi

$$L(\theta t + 1\ ) \le L(\theta t\ ) - \frac{\eta}{2}\|\nabla L(\theta t\ )\|^2$$

Sommando su t = 0,…,T−1 e usando la telescopicità:

$$\sum_{t=0}^{T-1} \|\nabla L(\theta t\ )\|^2 \le \frac{2}{\eta}\left[L(\theta_0) - L^*\right]$$

Il minimo della somma è raggiunto al tempo $t^* = \text{argmin}_t \|\nabla L(\theta_t)\|^2$, per cui

$$mint \le T\ \ \|\nabla L(\theta t\ )\|^2 \le \frac{2[L(\theta_0) - L^*]}{\eta T}$$

Questo dà il tasso O(1/T) per la norma al quadrato del gradiente — convergenza sub-lineare a un punto stazionario. ∎

12.2.2 Convergenza lineare per funzioni L-smooth e μ-fortemente convesse

Teorema 12.2 (Convergenza lineare di GD). *Se L è L-smooth e μ-fortemente convessa con numero di condizionamento κ = L/μ, allora con η = 1/L:*

$$\|\theta t \ - \ \theta * \|^2 \leq \left(1 \ - \ \frac{1}{\kappa}\right)^{t} \|\theta_0 \ - \ \theta * \|^2$$

La dimostrazione usa la co-coercività del gradiente per funzioni L-smooth e μ-fortemente convesse. Per tali funzioni vale la disuguaglianza

$$(\nabla L(\theta) \ - \ \nabla L(\theta'))^{\mathrm{T}}(\theta - \theta') \geq \frac{\mu L}{\mu + L} \|\theta - \theta'\|^2 \ + \ \frac{1}{\mu + L} \|\nabla L(\theta) - \nabla L(\theta')\|^2$$

Con θ' = θ*, ∇L(θ*) = 0, e applicando la discesa garantita:

$$\|\theta t + 1 \ - \ \theta * \|^2 \ = \ \|\theta t \ - \ \eta \nabla L(\theta t \ ) \ - \ \theta * \|^2$$

$$= \ \|\theta t \ - \ \theta * \|^2 \ - \ 2\eta \, \nabla L(\theta t \ )^{\mathrm{T}}(\theta t \ - \ \theta *) \ + \ \eta^2 \|\nabla L(\theta t \ )\|^2$$

Per μ-forte convessità: $\nabla L(\theta\_t)^{\mathrm{T}}(\theta\_t - \theta^*) \geq \mu \| \theta\_t - \theta^* \|^2 + L^{-1} \| \nabla L(\theta\_t) \|^2$. Con η = 1/L:

$$\|\theta t + 1 \ - \ \theta * \|^2 \leq \|\theta t \ - \ \theta * \|^2 \ - \ \frac{2\mu}{L} \|\theta t \ - \ \theta * \|^2 \ = \ \left(1 \ - \ \frac{2}{\kappa}\right) \|\theta t \ - \ \theta * \|^2$$

Iterando il contratto di contrazione $(1 - 2/\kappa) < 1$ si ottiene la convergenza geometrica dichiarata. Il numero di iterazioni per ε-accuratezza è $T = O(\kappa \log(1/\varepsilon))$. ∎

12.2.3 Stochastic Gradient Descent

Nell'addestramento di reti neurali con N esempi e mini-batch di dimensione B, il gradiente è stimato su B campioni casuali:

$$gt \ = \ \frac{1}{B} \sum_{i \in \mathcal{S}\_t} \nabla Li \ (\theta t \ )$$

dove $\mathcal{S}_t \subset \{1,\ldots,N\}$ è il mini-batch campionato uniformemente senza reinserimento, e $L_i(\theta) = \ell(f_\theta(x_i), y_i)$ è la perdita sull'i-esimo esempio. Il gradiente stocastico è non distorto: $\mathbb{E}[g_t] = \nabla L(\theta_t)$, ma ha varianza $\sigma^2 > 0$.

Teorema 12.3 (Convergenza SGD per funzioni L-smooth). *Sia L L-smooth e $\mathbb{E}[\|g_t - \nabla L(\theta_t)\|^2] \le \sigma^2$. Con learning rate $\eta_t = \eta_0/\sqrt{t}$ si ha:*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla L(\theta t\ )\|^2] \le O\left(\frac{\sigma}{\sqrt{T}}\right)$$

Il learning rate decrescente $\eta_t = \eta_0/\sqrt{t}$ è necessario per garantire sia che l'algoritmo converga ($\Sigma\ \eta_t = \infty$) sia che il rumore non impedisca la convergenza ($\Sigma\ \eta^2_t < \infty$) — le condizioni di Robbins-Monro. Tasso di convergenza $O(1/\sqrt{T})$ è il bound minimax ottimale per funzioni convesse con gradiente rumoroso (lower bound di Nemirovskii).

15.3 Adam: derivazione e analisi

Adam (Adaptive Moment Estimation, Kingma e Ba, 2015) è l'ottimizzatore de facto per l'addestramento di reti profonde. Combina il momentum del primo momento del gradiente con la scalatura adattiva del secondo momento, con correzioni di bias per i passi iniziali.

L'algoritmo mantiene stime del primo momento (media) e del secondo momento non centrato (varianza) del gradiente. Gli aggiornamenti sono:

$$mt \ = \ \beta_1\, mt - 1 \ + \ (1 - \beta_1)\, gt$$

$$vt \ = \ \beta_2\, vt - 1 \ + \ (1 - \beta_2)\, gt^2$$

dove la moltiplicazione g^2_t è element-wise. Con $\beta_1 = 0.9$ e $\beta_2 = 0.999$ (valori standard), i momenti decadono esponenzialmente.

12.3.1 Derivazione delle correzioni di bias

Il momento m_t è una media pesata dei gradienti passati. Se $m_0 = 0$, espandendo la ricorrenza:

$$mt \;=\; (1 - \beta_1) \sum_{i=1}^{t} \beta 1^{t-i}\, gi$$

Prendendo il valore atteso e assumendo stazionarietà $\mathbb{E}[g_i] = g$ (gradiente vero costante):

$$\mathbb{E}[mt\;] \;=\; g \cdot (1 - \beta_1) \sum_{i=1}^{t} \beta 1^{t-i} \;=\; g \cdot (1 - \beta t^{1})$$

dove si è usata la somma geometrica $\Sigma_{i=1}^{t} \beta_1^{t-i} = (1 - \beta_1^t)/(1 - \beta_1)$. Quindi $\mathbb{E}[m_t] = g \cdot (1 - \beta_1^t) \neq g$ per t piccolo: il momento sottostima il gradiente vero per effetto dell'inizializzazione zero. La correzione del bias per il primo momento è:

$$\hat{mt} \;=\; \frac{mt}{1 - \beta 1^{t}} \Longrightarrow \mathbb{E}[\hat{mt}\;] \;=\; g$$

Analogamente per il secondo momento v_t:

$$\mathbb{E}[vt\;] \;=\; (1 - \beta 2^{t}) \cdot \mathbb{E}[g^2]$$

cosicché la stima corretta è

$$\hat{vt} \;=\; \frac{vt}{1 - \beta 2^{t}} \Longrightarrow \mathbb{E}[\hat{vt}\;] \;=\; \mathbb{E}[g^2]$$

L'aggiornamento finale di Adam è:

$$\theta t + 1 \;=\; \theta t \;-\; \eta \cdot \frac{\hat{mt}}{\sqrt{\hat{vt}} + \varepsilon}$$

dove $\varepsilon \approx 10^{-8}$ garantisce stabilità numerica. La divisione element-wise per $\sqrt{\hat{v}_t} + \varepsilon$ scala il learning rate in modo adattivo: parametri con gradiente storicamente grande ricevono un passo piccolo, parametri con gradiente piccolo ricevono un passo grande. Questo equivale a un

precodizionamento diagonale approssimato con la matrice diagonale D = diag(√v̂_t + ε).

12.3.2 Analisi di convergenza di Adam

L'analisi di convergenza di Adam è più delicata di quella di SGD. Reddi et al. (2018) hanno mostrato che Adam nella sua forma originale può divergere su problemi convessi semplici, e hanno proposto AMSGrad come correzione. Il problema è che la stima del secondo momento v̂_t può diminuire nel tempo (se il gradiente recente è piccolo), causando un passo effettivo η/√v̂_t che aumenta anziché diminuire.

La variante AMSGrad introduce il massimo running dei secondi momenti:

$$\hat{v}^{max}t = max(\hat{v}^{max}t - 1 , \hat{v}t)$$

$$\theta t + 1 = \theta t - \eta \cdot \frac{\hat{m}t}{\sqrt{\hat{v}^{max}t} + \varepsilon}$$

Reddi et al. dimostrano per AMSGrad la convergenza con bound O(1/√T) per funzioni non convesse, identico a quello di SGD ma con convergenza empirica spesso più rapida.

Zaheer et al. (2018) hanno dimostrato che Adam con learning rate η = Θ(1/√T) converge a un punto ε-stazionario con T = O(σ²/ε^4) aggiornamenti totali, identico a SGD in termini di ordine ma con costanti migliori per dati non stazionari.

15.4 Batch Normalization: teoria e regolarizzazione del paesaggio

La Batch Normalization (BatchNorm, BN; Ioffe e Szegedy, 2015) è una delle tecniche più efficaci per accelerare l'addestramento di reti profonde e ridurre la sensibilità all'inizializzazione e al learning rate. La sua analisi

matematica rivela effetti profondi sulla geometria della superficie di perdita.

12.4.1 Definizione formale

Per un layer con attivazioni x ∈ ℝ^{B×d} (B = batch size, d = numero di feature), la BatchNorm normalizza ogni feature j = 1,…,d attraverso il batch:

$$\mu j = \frac{1}{B}\sum_{b=1}^{B} xbj$$

$$\sigma 2\ j = \frac{1}{B}\sum_{b=1}^{B}\left(xbj - \mu j\right)^2$$

$$\hat{x}bj = \frac{xbj - \mu j}{\sqrt{\sigma 2\ j + \varepsilon}}, \quad ybj = \gamma j\ \hat{x}bj + \beta j$$

dove γ_j, β_j ∈ ℝ sono i parametri di scala e traslazione apprendibili. All'inferenza, μ_j e σ²_j vengono rimpiazzati dalle statistiche medie di training calcolate con un'EMA (exponential moving average) sui batch di addestramento.

12.4.2 Analisi del gradiente attraverso BatchNorm

Il calcolo del gradiente attraverso BN richiede attenzione perché la normalizzazione accoppia tutti gli esempi nel batch. La derivata di L rispetto a x_{bj} è:

$$\frac{\partial L}{\partial xbj} = \frac{\gamma j}{\sqrt{\sigma 2\ j + \varepsilon}}\left(\frac{\partial L}{\partial \hat{y}bj} - \frac{1}{B}\sum_{b'}\frac{\partial L}{\partial \hat{y}b'j} - \frac{\hat{x}bj}{B}\sum_{b'}\hat{x}b'j\ \frac{\partial L}{\partial \hat{y}b'j}\right)$$

I due termini sottrattivi rappresentano l'accoppiamento introdotto dalla normalizzazione: il gradiente di un esempio dipende dai gradienti di tutti gli altri esempi nel batch. Per B → ∞, questi termini tendono a zero e il gradiente BN si riduce al gradiente senza BN scalato per γ_j/σ_j.

12.4.3 Effetto della BatchNorm sulla regolarizzazione del paesaggio

Santurkar et al. (2019) hanno pubblicato un'analisi fondamentale che confuta la spiegazione originale di Ioffe e Szegedy ("riduzione del covariate shift interno") e propone una spiegazione basata sulla geometria del paesaggio. I risultati principali sono:

> **Teorema 12.4 (Santurkar et al., 2019).** *Sia L_BN la perdita di una rete con BatchNorm. Il gradiente di L_BN è β-Lipschitz con β sensibilmente inferiore a β della rete senza BN: $\|\nabla L\_BN(\theta) - \nabla L\_BN(\theta')\| \leq \beta\_BN \|\theta - \theta'\|$ con $\beta_BN \ll \beta$_senza-BN.*

La dimostrazione calcola esplicitamente le costanti di smoothness con e senza BN. Per un layer lineare y = Wx + b con BN applicata sull'output:

$$\partial LBN\ /\partial W\ =\ \frac{\partial L}{\partial \hat{y}} \cdot \left(I\ -\ \frac{11^{\mathrm{T}}}{B}\ -\ \hat{x}\hat{x}^{\mathrm{T}}/B\right) \cdot \hat{x}/\sigma(y)$$

dove la norma di questo gradiente è limitata da $\|\gamma\|\_\infty$ / σ_min(BN). La normalizzazione a σ_j > 0 limita la Lipschitz constant del gradiente, rendendo la superficie di perdita più uniformemente liscia.

Invarianza ai riscalamenti dei pesi. Una proprietà chiave della BatchNorm è l'invarianza ai riscalamenti dei pesi: per un layer W, la normalizzazione garantisce che la funzione computata dal layer sia invariante alla sostituzione W → cW per c > 0. Formalmente, se y = BN(Wx), allora BN(cWx) = BN(Wx) per ogni c > 0. Questa proprietà ha importanti conseguenze:

Il gradiente rispetto ai pesi è limitato: $\|\partial L/\partial W\| \leq \lambda$_max per una costante λ_max indipendente dalla norma di W.

L'ottimizzatore opera effettivamente sulla direzione dei pesi anziché sulla loro norma. L'iperspazio {W : ‖W‖ = c} è una varietà su cui la Loss è costante (per ogni c > 0), quindi la dinamica di ottimizzazione è essenzialmente proiettata su questa varietà.

Proposizione 12.1 (Invarianza e gradiente efficace). *Per W tale che y = BN(Wx), si ha: (i) ∂L/∂W = P_{W}⊥ [∂L/∂W], dove P_{W}⊥ è la proiezione ortogonale al sottospazio span(W); (ii) ‖∂L/∂W‖_F ≤ ‖∂L/∂W‖_F · (1 − ‖W‖²_F / ‖W‖²_F) = 0 nella direzione di W, cioè il gradiente effettivo è ortogonale a W.*

12.4.4 BatchNorm vs. LayerNorm: confronto analitico

LayerNorm (§10.5) e BatchNorm differiscono nell'asse di normalizzazione: LN normalizza attraverso le feature per ogni esempio, BN normalizza attraverso il batch per ogni feature. Questa differenza ha implicazioni profonde:

Per batch size B→∞, BN converge a una normalizzazione con statistiche empiriche globali, mentre LN è indipendente dal batch. Questo rende LN preferibile per modelli con sequenze di lunghezza variabile (Transformer) e LN ha gradiente stabile per ogni B.

Per le reti convoluzionali con feature spaziali, BN aggrega statistiche su tutte le posizioni spaziali nel batch, producendo una smoothness del paesaggio proporzionale a √(B·H·W) dove H, W sono le dimensioni spaziali. LN aggrega solo su canali, producendo effetti diversi.

15.5 Learning Rate Scheduling e Warm-up

Un elemento fondamentale dell'addestramento pratico di modelli profondi è la schedulazione del learning rate. Nella pratica dei Transformer, si usa quasi universalmente il cosine annealing con warm-up lineare iniziale:

$$\eta(t) = \begin{cases} \eta_{min} + \frac{t}{t_{warmup}} (\eta_{max} - \eta_{min}) & per\ t \le t_{warmup} \\ \eta_{min} + \frac{1 + cos(\pi(t - t_{warmup})/(T - t_{warmup}))}{2} (\eta_{max} - \eta_{min}) & per\ t > t_{warmup} \end{cases}$$

Il warm-up è analiticamente motivato dall'analisi di Liu et al. (2019): all'inizio dell'addestramento, la stima del secondo momento v_t in Adam non è ancora stabile (inizializzata a 0, richiede ≈ $1/(1-\beta_2) = 1000$ passi per convergere), causando varianza elevata nell'aggiornamento. Il warm-up riduce η durante questa fase instabile.

Il cosine annealing (Loshchilov e Hutter, 2017) fornisce una diminuzione progressiva del learning rate che, contrariamente al decadimento polinomiale, non porta mai a $\eta = 0$, mantenendo la capacità di fine-tuning.

15.6 Convergenza dell'Adam per l'addestramento di Transformer

L'analisi combinata di Adam con Pre-Norm residual e LayerNorm, rilevante per i Transformer, porta al seguente quadro teorico sintetico.

Per una rete Transformer con L strati, Pre-Norm residual e LN, la smoothness del gradiente è L-smooth con

$$L = O\left(\frac{\|\gamma\|^2_\infty}{\varepsilon_{LN}} \cdot L_{attn} \cdot L\right)$$

dove $L_{attn} \leq n$ (lunghezza sequenza) è la costante di Lipschitz della softmax, e L è la profondità. Questo spiega perché il learning rate ottimale scala approssimativamente come $\eta^* = O(1/\sqrt{(L \cdot n)})$.

L'analisi di Tian et al. (2023) mostra che per Transformer addestrati con AdamW su testi naturali, la traiettoria di ottimizzazione attraversa le seguenti fasi: (i) rapid alignment dell'embedding space alle posizioni sintatticamente simili; (ii) specializzazione delle teste di attenzione su relazioni specifiche; (iii) apprendimento delle FFN come lookup table per fattualità. Questa sequenza è coerente con le osservazioni empiriche di phase transitions nell'emergenza di abilità.

15.7 Geometria dell'ottimizzazione per il Reinforcement Learning

Il paesaggio di perdita in RL differisce fondamentalmente da quello supervisionato: la distribuzione dei dati (stati visitati) dipende dalla politica corrente, introducendo non-stazionarietà. Questo produce una superficie di perdita non fissa — il target si sposta mentre il modello si aggiorna.

Per PPO, l'obiettivo clippeato L^{CLIP}(θ) ha una proprietà fondamentale: il suo gradiente ∇_θ L^{CLIP}(θ) è sempre bounded quando r_t(θ) ∈ [1−ε, 1+ε], anche se il gradiente dell'obiettivo non clippeato potrebbe esplodere. Formalmente:

$$\|\nabla\theta \ \ L \ \ ^{CLIP}(\theta)\| \leq \ \ \ \frac{1}{1-\varepsilon} \ \|\hat{A}\|\_\infty \cdot \|\nabla\theta \ \ log\, \pi\theta \ \ (a|s)\|$$

Il bound dipende da ‖Â‖_∞ (il massimo del vantaggio nel batch) e da ‖∇ log π_θ‖ (la sensibilità della politica ai parametri). Il clipping limita implicitement la curvatura dell'obiettivo nella direzione dei parametri, agendo come un regolarizzatore geometrico sulla superficie di perdita di RL.

PARTE III

World Models

Questa Parte espone in modo rigoroso la teoria e le architetture dei modelli del mondo appresi, tracciando il percorso dalle origini cognitive e computazionali degli anni Novanta fino ai sistemi odierni che dominano benchmark eterogenei con una singola configurazione di iperparametri. Ogni affermazione quantitativa è ricavata dai paper allegati; le derivazioni matematiche sono condotte dal principio primo senza omettere i passaggi non ovvi.

Capitolo 16 — Cos'è un World Model

16.1 Definizione formale e struttura del POMDP

Un ***world model*** è un modello probabilistico parametrico θ che approssima la distribuzione generativa congiunta di un processo decisionale di Markov parzialmente osservabile. Il POMDP di riferimento è la settupla $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, \Omega, \gamma)$ dove: $\mathcal{S}$ è lo spazio degli stati fisici latenti; $\mathcal{A} \subseteq \mathbb{R}^{d_a}$ è lo spazio delle azioni continue; $\mathcal{O} \subseteq \mathbb{R}^{d_o}$ è lo spazio delle osservazioni sensoriali — tipicamente frame RGB 64×64×3; $T: \mathcal{S} \times \mathcal{A} \to \mathcal{R}(\mathcal{S})$ è la funzione di transizione stocastica; $R: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ è la funzione di ricompensa scalare; $\Omega: \mathcal{S} \to \mathcal{R}(\mathcal{O})$ è il modello di emissione; $\gamma \in (0, 1)$ è il fattore di sconto intertemporale.

Un world model impara congiuntamente quattro componenti. Il ***modello di rappresentazione*** (encoder ammortizzato) $q_\theta(s_t \mid h_t, x_t)$ comprime l'osservazione x_t nell'informazione aggiornata h_t della storia. Il ***predittore dinamico*** (prior) $p_\theta(\hat{s}_t \mid h_t)$ anticipa il prossimo stato senza accedere all'osservazione — realizzando la simulazione interna. Il ***decoder*** $p_\theta(\hat{x_t} \mid s_t)$ ricostruisce l'osservazione dal latente, fornendo il segnale di apprendimento. Il ***predittore di ricompensa*** $p_\theta(\hat{r_t} \mid s_t)$ produce la ricompensa stimata, necessaria per l'ottimizzazione della politica.

Lo stato del modello è la coppia deterministico-stocastica: $s\_t = \{h\_t, z\_t\}$, dove $h\_t \in \mathbb{R}\hat{}\{d\_h\}$ è la componente deterministica della GRU e z_t è la componente stocastica (gaussiana o categoriale a seconda della versione).

16.2 Radici storiche: da Tolman a Schmidhuber

La nozione di mappa cognitiva interna risale a Edward Tolman (1948), che dimostrò sperimentalmente come i ratti in labirinto costruiscano rappresentazioni spaziali latenti indipendenti dal comportamento diretto

— la cosiddetta ***latent learning***. Il framework della ***predictive coding*** (Rao & Ballard 1999) ha poi formalizzato l'idea in termini computazionali: la corteccia genera costantemente predizioni top-down sull'input sensoriale e propaga solo il residuo (prediction error) ai livelli superiori. Karl Friston ha successivamente esteso questo paradigma nel ***Free Energy Principle***: ogni sistema biologico minimizza una variante dell'ELBO variazionale — la free energy — che misura la sorpresa predittiva del modello interno rispetto ai dati sensoriali. Questo principio è formalmente equivalente all'obiettivo di addestramento del world model computazionale.

Jürgen Schmidhuber, nel manoscritto ***Making the World Differentiable*** (1990), propose di addestrare una RNN affinché funzionasse come simulatore differenziabile dell'ambiente, separata dal controllore che genera le azioni. Il contributo fondamentale fu riconoscere che, se il modello del mondo è differenziabile, il gradiente della ricompensa può essere propagato analiticamente attraverso di esso — esattamente il meccanismo che Dreamer implementerà trent'anni dopo. Lo stesso Schmidhuber introdusse la ***curiosità intrinseca*** basata sul modello: ricompensare l'agente proporzionalmente al *guadagno di informazione* che una nuova osservazione produce nel modello — un precursore diretto dei metodi di esplorazione basati sull'errore di predizione.

16.3 PILCO e il decennio dei Gaussian Process (2000–2013)

L'approccio di PILCO (Deisenroth & Rasmussen 2011) impiega un Gaussian Process (GP) come modello delle dinamiche, valorizzando la capacità dei GP di quantificare l'incertezza epistemica in forma analitica. Dato un dataset di transizioni $\{(s_i, a_i, s'_i)\}$, il GP definisce una distribuzione sulle funzioni:

$$f \sim GP(m(s,a),\ \ k((s,a),(s',a')))(1)$$

La previsione a un passo per un nuovo input (s, a) è gaussiana con media e varianza:

$$\mu_* = K(x*, X)(K(X,X) + \sigma n^2 I) - 1\ y,\ \sigma^2 * = k(x*, x*) - K(x*, X)(K + \sigma n^2 I) - 1$$

(2)

Il limite strutturale di PILCO è la complessità $O(n^3)$ nella dimensione del dataset n, che ne impedisce l'applicazione a osservazioni visive ad alta dimensionalità. PILCO risolve task di controllo a bassa dimensionalità (pendolo, cartpole) con decine di episodi, ma la scalabilità a manipolazione visiva rimane preclusa fino all'avvento dei VAE profondi.

16.4 La svolta del 2018: World Models di Ha e Schmidhuber

Ha e Schmidhuber (2018) dimostrano praticamente la fattibilità dell'architettura Vision-Memory-Controller su due ambienti OpenAI Gym: CarRacing-v0 (che nessun altro agente aveva risolto all'epoca con punteggio ≥ 900) e VizDoom. Il risultato più sorprendente è l'addestramento del controllore ***interamente nel dream*** — nelle traiettorie generate dal modello con temperatura $\tau = 1.15$ — con successivo trasferimento nell'ambiente reale senza ulteriori aggiustamenti, ottenendo un punteggio di 906 ± 21 su CarRacing. La temperatura elevata crea un ambiente di sogno più caotico che induce comportamenti robusti alle imprecisioni del modello.

16.5 Tassonomia Formale: World Model Generativi e Predittivi

La distinzione tra world model generativi e world model predittivi nel latente è la linea di faglia più importante nella tassonomia delle architetture analizzate in questa monografia. Un world model generativo apprende la distribuzione congiunta p(o_{t+1} | o_{1:t}, a_{1:t}) sulle

osservazioni future nello spazio originale (pixel, audio, sensori): produce previsioni che possono essere confrontate direttamente con l'esperienza sensoriale, al costo di modellare tutta la varianza irrilevante del segnale (texture, illuminazione, rumore). Un world model predittivo nel latente apprende invece la distribuzione $p(z_{t+1} \mid z_{1:t}, a_{1:t})$ su rappresentazioni compresse $z_t = \varphi(o_t) \in Z$, dove φ è un encoder addestrato congiuntamente al modello di transizione.

Formalmente, un world model è una tripla (Z, T, R) dove Z è lo spazio degli stati latenti, $T: Z \times A \to \Delta(Z)$ è il modello di transizione (distribuzione sullo stato successivo dato lo stato corrente e l'azione), e $R: Z \times A \to \mathbb{R}$ è il modello di ricompensa. Questa struttura è identica a quella di un MDP con spazio degli stati Z (si veda il Capitolo 11 per la teoria degli MDP): il world model è un MDP appreso dall'esperienza, sul quale si può eseguire la pianificazione senza interagire con l'ambiente reale. La qualità del world model si misura con due criteri: la fidelity (quanto accuratamente T replica la dinamica reale) e l'abstraction (quanto la struttura di Z facilita la pianificazione rispetto allo spazio delle osservazioni originali).

16.6 Il Ruolo dell'Azione: World Model Passivi e Attivi

Una distinzione fondamentale separa i world model passivi, che modellano la dinamica del mondo indipendentemente dalle azioni dell'agente, dai world model attivi (action-conditioned), che modellano esplicitamente come le azioni modificano la traiettoria degli stati. I world model passivi (es. modelli di previsione video come VideoGPT, MCVD) sono addestrati su video senza informazioni sulle azioni e non possono essere usati per il planning di controllo; i world model attivi come RSSM (sezione 17.4) e DreamerV3 (sezione 17.7) condizionano la transizione sull'azione a_t e sono la base per il Model-Based Reinforcement Learning e per il controllo predittivo neurale del Capitolo 27.

La struttura matematica del world model attivo è un Modello Markoviano di Controllo (Controlled Markov Model): (S, A, T, R, γ) dove T(s' | s, a) è la distribuzione di transizione condizionata sull'azione. Quando S = Z (spazio latente appreso) e T è implementata da una rete neurale ricorrente o da un Transformer, si ottiene il paradigma dei world model neurali che è il filo conduttore della Parte III. La relazione tra la struttura del world model e la struttura dell'architettura JEPA (Parte IV) emerge dal confronto: JEPA apprende la mappa z_{t+1} = Pred(z_t, a_t) senza ricostruzione esplicita, mentre RSSM apprende p(z_{t+1} | z_t, a_t, h_t) come distribuzione stocastica condizionata sullo stato ricorrente h_t. Il Capitolo 25 formalizza questa unità strutturale tramite il framework LPA (Latent Predictive Architecture).

Capitolo 17 — Architetture

17.1 VAE: encoder, decoder e obiettivo variazionale

Il Variational Autoencoder (Kingma & Welling 2013, Rezende et al. 2014) è il componente Vision dell'architettura di Ha e Schmidhuber. L'encoder produce i parametri della distribuzione posterior:

$$q\theta\ (zt\ |ot\ )\ =\ \mathcal{N}(\mu\theta\ (ot\ ),\ diag(\sigma 2\ \theta\ (ot\ )))(3)$$

Il campionamento avviene tramite il reparameterization trick (dimostrato nel §12.5):

$$zt\ \ =\ \mu\theta\ (ot\ )\ +\ \sigma\theta\ (ot\ )\odot\varepsilon,\quad \varepsilon\sim\mathcal{N}(0,\ I32\ )(4)$$

Il decoder $p_\varphi(o_t \mid z_t)$ è implementato con quattro strati ConvTranspose2D e produce la ricostruzione $\hat{o}_t$. La funzione di perdita del VAE è l'ELBO (derivato interamente nel §12.2–12.3):

$$\mathcal{L}VAE\ \ =\ \mathbb{E}q\ [ln\,p\varphi\ (ot\ |zt\ )]\ -\ KL[q\theta\ (zt\ |ot\ )\parallel\mathcal{N}(0,I)](5)$$

17.2 MDN-RNN: mixture density network per la memoria temporale

Il modulo Memory di Ha-Schmidhuber è un'LSTM con output a Mixture Density Network (Bishop 1994). L'LSTM aggiorna il suo stato nascosto come:

$$ht\ ,\ ct\ \ =\ LSTM(ht\ -\ 1\ ,\ ct\ -\ 1\ ;\ [zt\ ,\ at\ ])(6)$$

Lo strato di output lineare produce i parametri di K gaussiane:

$$(\pi 1{:}K\ ,\ \mu 1{:}K\ ,\ log\,\sigma 1{:}K\ )\ =\ Wout\ \ ht\ \ +\ bout\ (7)$$

La distribuzione predittiva è la mixture:

$$p(zt + 1 \mid at , zt , ht) = \Sigma k = 1 K \pi k \mathcal{N}(zt + 1 ; \mu k , \sigma 2 k I) \quad (8)$$

La log-verosimiglianza negativa della mixture è la funzione di perdita:

$$\mathcal{L}MDN = - ln \Sigma k \pi k (ht) \mathcal{N}(zt + 1 ; \mu k (ht), \sigma 2 k (ht) I) \quad (9)$$

Il parametro di temperatura τ scala le varianze durante la generazione del dream: $\tilde{\sigma_k}(\tau) = \tau \sigma_k$. Con τ = 1.15, l'ambiente sognato è più stocastico dell'ambiente reale, inducendo robustezza nel controllore.

17.3 Controller CMA-ES

Il controller è un modello lineare:

$$at = Wc [zt ; ht] + bc \quad (10)$$

I parametri (W_c, b_c) sono ottimizzati con CMA-ES (Hansen 2001). CMA-ES mantiene una distribuzione gaussiana sull'insieme dei parametri $\theta \sim \mathcal{N}(m^{(i)}, \sigma^{(i)2} C^{(i)})$ e aggiorna media, passo-scala e matrice di covarianza dopo aver valutato λ individui. La regola di aggiornamento CMA-ES include il rank-one update:

$$C(i + 1) = (1 - c1 - c\mu) C(i) + c1 pc T pc + c\mu \Sigma j = 1 \mu wj yj T yj \quad (11)$$

dove p_c è il percorso evolutivo cumulato e $y_j = (x_j - m^{(i)}) / \sigma^{(i)}$. Per il controllore di CarRacing con 867 parametri, 16 worker CPU eseguono rollout paralleli nel dream, riducendo il tempo di training a circa un'ora.

17.4 RSSM: Recurrent State Space Model

11.4.1 Struttura ibrida deterministico-stocastica

L'RSSM (Hafner et al. 2018, PlaNet) suddivide lo stato latente in due componenti complementari, risolvendo il compromesso tra espressività e addestrabilità. La componente ***deterministica*** h_t scorre come BPTT puro, consentendo la propagazione di gradienti su lunghi orizzonti. La componente ***stocastica*** z_t cattura l'incertezza residua che la sola storia deterministica non risolve. Le equazioni di aggiornamento sono:

Seq: $ht = GRU(ht - 1 , zt - 1 , at - 1)$(12)

Posterior: $zt \sim q\phi (zt | ht , xt)$(13)

Prior: $\hat{z}t \sim p\phi (\hat{z}t | ht)$(14)

Decoder: $\hat{x}t \sim p\phi (\hat{x}t | ht , zt)$(15)

Reward: $\hat{r}t \sim p\phi (\hat{r}t | ht , zt)$(16)

La proprietà cruciale è che la fase di immaginazione usa solo il prior (eq. 14): non richiede l'osservazione reale x_t. La GRU propaga il gradiente ininterrottamente attraverso h_1, h_2, …, h_H, rendendola la backbone della retropropagazione attraverso i sogni. Hafner et al. (2019, DreamerV1) mostrano sperimentalmente che rimozione della componente deterministica riduce il punteggio medio da 823 a 311 su DMControl.

11.4.2 PlaNet: pianificazione CEM sull'RSSM

PlaNet seleziona le azioni tramite Cross-Entropy Method (CEM). La distribuzione sui piani di azione è inizializzata come gaussiana isotropa:

$p(a1: H) = \Pi t = 1 H \mathcal{N}(at ; \mu(0) t , (\sigma(0) t)2 I)$(17)

Ad ogni iterazione i = 1 … I:

$$at, j \ \sim \mathcal{N}(\mu(i)\ t\ ,\ (\sigma(i)\ t\ )2\ \ I), \quad j = 1...J(18)$$

$$Rj \ = \ \Sigma\tau = 1\ H\ \ \gamma\tau - 1\ \ \hat{r}\tau\ (a1{:}\,H, j\ ) \quad calcolato\ con\ il\ prior\ del\ modello$$

(19)

$$\mu(i + 1)\ t \ = meanj{\in}\mathcal{K}\ \ (at, j\ ), \quad \sigma(i + 1)\ t \ = stdj{\in}\mathcal{K}\ \ (at, j\ )$$

(20)

dove $\mathcal{K}$ è l'insieme dei $K = \lceil J/10 \rceil = 100$ piani con il maggior ritorno simulato (top-10%). PlaNet usa $J = 1000$, $H = 12$, $I = 10$ ottenendo con 5×10^6 passi un punteggio medio di 332 su DMControl — superiore di 10× all'A3C model-free con stati proprietocettivi, con un costo di diecimila forward pass del modello per ogni singola azione selezionata.

17.5 Dreamer v1: actor-critic differenziabile

Dreamer (Hafner et al. 2020, ICLR) sostituisce CEM con una coppia actor-critic addestrata interamente nell'immaginazione. Il modello attore genera azioni continue come gaussiana trasformata con tanh:

$$a\tau \ = \ tanh(\mu\varphi\ (s\tau\ ) \ + \ \sigma\varphi\ (s\tau\ )\,\varepsilon), \quad \varepsilon \sim \mathcal{N}(0, I)(21)$$

I λ-return bilanciano bias e varianza delle stime di valore. La stima a k passi è:

$$Vk\ N\ (s\tau\ ) \ = \ \mathbb{E}[\Sigma n = \tau\ h - 1\ \ \gamma n - \tau\ \ rn \ \ + \ \gamma h - \tau\ \ v\psi\ (sh\ )], \quad h \ = \ min(\tau + k,$$

(22)

Il λ-return è la media geometricamente pesata:

$$V\lambda\ (s\tau\ )\ =\ (1\ -\ \lambda)\ \Sigma n\ =\ 1\ H\ -\ 1\ \ \lambda n\ -\ 1\ \ Vn\ N\ (s\tau\ )\ +\ \lambda H\ -\ 1\ \ VH\ N\ (s\tau\ )$$

(23)

L'obiettivo dell'attore è massimizzare il ritorno atteso lungo le traiettorie immaginate, propagando il gradiente analiticamente attraverso le predizioni del modello tramite reparameterization:

$$max\varphi\ \ \mathbb{E}q\ [\Sigma\tau\ =\ t\ t\ +\ H\ \ V\lambda\ (s\tau\ )](24)$$

Il critico minimizza l'errore quadratico sui target con stop-gradient:

$$min\psi\ \ \mathbb{E}[\Sigma\tau\ =\ t\ t\ +\ H\ \ ½\ (v\psi\ (s\tau\ )\ -\ sg(V\lambda\ (s\tau\ )))2\ ](25)$$

Dreamer raggiunge un punteggio medio di 823 su 20 task visivi del DMControl Suite con 5×10^6 passi, superando D4PG (786 punti con 10^8 passi). Il guadagno di efficienza campionaria è di circa 20× rispetto al miglior algoritmo model-free.

17.6 DreamerV2: rappresentazioni categoriali e KL balancing

DreamerV2 (Hafner et al. 2021) sostituisce il latente gaussiano con un vettore di N = 32 variabili categoriali, ciascuna con K = 32 classi. Lo spazio latente discreto ha 32^{32} combinazioni possibili, con dimensione effettiva 32 × 32 = 1024 valori. Il vantaggio delle rappresentazioni discrete è la nitidezza per stati a confini netti (sprite Atari, bordi di oggetti), e la compatibilità con il Transformer autoregressivo.

11.6.1 Straight-through estimator

Il campionamento one-hot non è differenziabile rispetto ai logit. Lo ***straight-through estimator*** (Bengio et al. 2013) approssima il gradiente dell'argmax come se fosse la softmax, implementando:

$$zt \;=\; one\_hot(argmax(lt\;)) \;+\; sg(softmax(lt\;) \;-\; one\_hot(argmax(lt\;))) \quad (26)$$

Nel forward pass il termine sg(·) è zero, quindi z_t è il corretto one-hot. Nel backward pass, sg azzera il gradiente di one_hot(argmax) e il flusso avviene attraverso softmax(l_t), che è differenziabile. Questo produce un gradiente biased (perché approssima argmax con softmax) ma a bassa varianza rispetto a REINFORCE.

11.6.2 KL balancing

La KL tra due categoriali q e p si decompone in cross-entropia meno entropia:

$$KL(q \,\|\, p) \;=\; \Sigma k \;\; qk \;\; ln(qk \;/pk\;) \;=\; H(q,p) \;-\; H(q) \quad (27)$$

Il KL balancing applica stop-gradient asimmetrico con coefficiente $\alpha = 0.8$:

$$\mathcal{L}KL \;\;=\; \alpha \cdot H(sg(q),\; p) \;+\; (1 \;-\; \alpha) \cdot H(q,\; sg(p)) \quad (28)$$

Il primo termine addestra il prior p ad avvicinarsi al posterior ($\alpha = 0.8$ gli assegna peso maggiore). Il secondo addestra il posterior q a diventare più predicibile dal prior. I free bits clippano il KL dal basso:

$$\mathcal{L}KL, freebits \;\;=\; max(\delta,\; \mathcal{L}KL\;), \quad \delta \;=\; 1\; nat \approx 1.44\; bit \quad (29)$$

17.7 DreamerV3: robustezza cross-domain

11.7.1 Symlog e symexp twohot

La funzione ***symlog*** comprime le grandi magnitudini mantenendo la linearità nell'origine:

$$symlog(x) = sign(x) \cdot ln(|x| + 1), \quad symexp(x) = sign(x) \cdot (e|x| - 1)$$
(30)

Per il predittore di ricompensa e il critico, DreamerV3 usa i ***bin a spaziatura esponenziale***:

$$B = \{ bi \} = symexp(linspace(- 20, + 20, 255))$$(31)

Il target continuo y viene codificato con ***twohot encoding***: trovati b_k e b_{k+1} più vicini a symlog(y), i pesi sono:

$$twohot(y)i = (bk + 1 - symlog(y))/(bk + 1 - bk) \; [i = k]$$
(32)

$$twohot(y)i = (symlog(y) - bk)/(bk + 1 - bk) \; [i = k + 1], \quad 0 \; altrimenti$$
(33)

La perdita è la cross-entropia categoriale sui soft target:

$$\mathcal{L}twohot \; (y) = - twohot(y)T \; log \; softmax(f(x, \theta))$$(34)

Il gradiente di questa perdita rispetto ai parametri della rete non dipende dalla magnitudine assoluta di y, ma solo dalla classificazione nei bin — il che disaccoppia la scala dei gradienti dalla scala delle ricompense.

11.7.2 Return normalization e critico distribuito

La normalizzazione dei ritorni usa i percentili al 5° e al 95°:

$$S = EMA(Per(R\lambda \; t \; , 95) - Per(R\lambda \; t \; , 5), \; 0.99)$$(35)

$$\bar{A}t = At \; / max(1, S), \quad At = R\lambda \; t - v\psi \; (st \;)$$(36)

Il critico stima la distribuzione completa dei ritorni su 255 bin, leggendo il valore come media pesata: $v_\psi(s_t) = softmax(f_\psi(s_t))^T \cdot B$. Per stabilizzare il

training, il critico si regolarizza verso la propria EMA — un target network soft:

$$\psi\_target \leftarrow \tau \cdot \psi\_target + (1 - \tau) \cdot \psi, \quad \tau = 0.99 \quad (37)$$

DreamerV3 è il primo algoritmo a raccogliere diamanti in Minecraft senza dati umani né curricula. Su 8 domini (Atari, ProcGen, DMLab, Proprio Control, Visual Control, BSuite, Minecraft, Atari100k) con iperparametri fissi, supera PPO su tutti i benchmark e MuZero su Atari con una frazione delle risorse.

11.7.3 Perdita composita del world model

$$\mathcal{L}(\phi) = \mathbb{E}[\Sigma t \, (\beta pred \; \mathcal{L}pred + \beta dyn \; \mathcal{L}dyn + \beta rep \; \mathcal{L}rep \,)] \quad (38)$$

$$\mathcal{L}pred = -\tfrac{1}{2} \| symlog(xt) - \hat{x}t \|2 - \ln p(rt \,|st) - \ln p(ct \,|st) \quad (39)$$

$$\mathcal{L}dyn = max(1, KL[\, sg(q\phi \; (zt \,|ht, xt)) \,\|\, p\phi \; (zt \,|ht) \,]) \quad (40)$$

$$\mathcal{L}rep = max(1, KL[\, q\phi \; (zt \,|ht, xt) \,\|\, sg(p\phi \; (zt \,|ht)) \,]) \quad (41)$$

17.8 IRIS: Transformer autoregressivo come world model

IRIS (Micheli, Alonso, Fleuret, ICLR 2023) interpreta il problem della modellazione delle dinamiche come un problema di ***sequence modeling su un vocabolario appreso***. L'encoder VQ-VAE mappa ogni frame $x_t \in \mathbb{R}^{64\times64\times3}$ in K = 16 token discreti da un codebook di dimensione |C| = 1024. Formalmente, il k-esimo token è:

$$zk\ t = argmini \, \|yk\ t - ei\|2, \quad yk\ t = CNN - enc(xt) \quad (42)$$

La perdita del VQ-VAE è la combinazione di ricostruzione, commitment e perceptual loss:

$$\mathcal{L}VQ = \|xt - dec(zt)\|L1 + \|sg(y), e\|2 + \beta\|y, sg(e)\|2 + \mathcal{L}perceptual \quad (43)$$

Il Transformer autoregressivo G (architettura GPT) modella la sequenza interleaved di token di frame e azioni. Per ogni passo temporale t, le tre distribuzioni predette da G sono:

$$\hat{z}k\ t+1 \sim pG\ (\hat{z}k\ t+1 \mid z{\leq}t\ ,\ a{\leq}t\ ,\ z{<}\ k\ t+1\ ) \quad (44)$$

$$\hat{r}t \sim pG\ (\hat{r}t \mid z{\leq}t\ ,\ a{\leq}t\ ) \quad (45)$$

$$\hat{d}t \sim pG\ (\hat{d}t \mid z{\leq}t\ ,\ a{\leq}t\ ) \quad (46)$$

Con il Transformer, IRIS ottiene uno human normalized score medio di 1.046 su Atari 100k (400.000 frame, ~2 ore di gameplay), superando gli umani in 10/26 giochi. Rispetto a DreamerV2 (che usa 200M frame per Atari), IRIS richiede 500× meno interazioni.

17.9 MuDreamer: world model predittivo senza ricostruzione

MuDreamer (Burchi & Timofte 2024) identifica la debolezza strutturale della perdita di ricostruzione pixel-wise: il world model deve modellare dettagli visivi irrilevanti per il task (texture di sfondo, illuminazione), riducendo la capacità disponibile per la codifica degli elementi task-relevant. Con distrattori visivi (sfondo sostituito con video naturali), DreamerV3 degrada significativamente. MuDreamer mantiene l'architettura RSSM ma aggiunge:

Predittore di valore $\hat{v}_t$ che predice il λ-return corrente dal modello nascosto; ***Predittore di azione*** $\hat{a}_{t-1}$ che, data l'osservazione corrente e lo

stato precedente, inferisce l'azione che ha prodotto il cambiamento — obbligando le rappresentazioni a codificare le cause del movimento piuttosto che il contenuto visivo non strutturato.

La perdita di predizione di MuDreamer è:

$$\mathcal{L}pred = -\ln p(rt\ |st\ ) - \ln p(ct\ |st\ ) - \ln p(R\lambda\ t\ |st\ ) - \ln p(at - 1\ |xt\ , st$$

(47)

La ***batch normalization*** nella rete di rappresentazione è cruciale per prevenire il learning collapse in assenza di ricostruzione: senza normalizzazione, il modello converge a stati costanti (collapse). MuDreamer supera DreamerV3 sui distrattori visivi del DMControl Suite e ottiene prestazioni comparabili su Atari 100k con training più rapido.

Capitolo 18 — Matematica dei Latent Dynamics Models

18.1 Inferenza variazionale e ELBO: derivazione completa

L'obiettivo fondamentale è massimizzare la log-verosimiglianza marginale:

$$ln\, p\theta\ (x1{:}T\ )\ =\ ln \int p\theta\ (x1{:}T\ ,\ s1{:}T\ )\, ds1{:}T\ (48)$$

Questa integrale è intrattabile. Moltiplicando e dividendo per $q_\theta(s_{1:T} \mid x_{1:T})$:

$$=\ ln\, \mathbb{E}q\ [\tfrac{p(x,s)}{q(s|x)}](49)$$

> ***Dimostrazione — Disuguaglianza di Jensen → ELBO***

La funzione $f(u) = ln\ u$ è strettamente concava su $\mathbb{R}^+$. La disuguaglianza di Jensen afferma che per ogni variabile aleatoria $U > 0$:

$$f(\mathbb{E}[U]) \geq \mathbb{E}[f(U)],\quad con\ uguaglianza \Leftrightarrow U\ =\ const\ q.c.(50)$$

Ponendo $U = p(x,s)/q(s \mid x)$, si ha:

$$ln\, \mathbb{E}q\ [\tfrac{p(x,s)}{q(s|x)}] \geq \mathbb{E}q\ [ln\tfrac{p(x,s)}{q(s|x)}](51)$$

Il secondo membro si sviluppa come:

$$\mathbb{E}q\ [ln\, p(x,s)\ -\ ln\, q(s|x)](52)$$

$$=\ \mathbb{E}q\ [ln\, p(x|s)]\ +\ \mathbb{E}q\ [ln\, p(s)]\ -\ \mathbb{E}q\ [ln\, q(s|x)](53)$$

$$=\ \mathbb{E}q\ [ln\, p(x|s)]\ -\ KL[q(s|x)\ \|\ p(s)]\ \ =:\ ELBO(\theta)(54)$$

Il gap tra log-verosimiglianza e ELBO è la KL:

$$\ln p(x) \; - \; ELBO \; = \; KL[q(s|x) \parallel p(s|x)] \geq 0 (55)$$

Prova: $\ln p(x) - ELBO = \mathbb{E}_q[\ln q(s|x) - \ln p(s|x)] = KL[q \parallel p(\cdot|x)] \geq 0$ per la non-negatività della KL (disuguaglianza di Gibbs).

□

18.2 Decomposizione step-by-step dell'ELBO per l'RSSM

Il modello generativo dell'RSSM si fattorizza come:

$$p\theta \; (x1{:}T \; , s1{:}T \;) \; = \; p(s1 \;) \, \Pi t \; = \; 2 \; T \;\; p\theta \; (st \; |ht \;) \cdot \Pi t \; = \; 1 \; T \;\; p\theta \; (xt \; |st \;)$$

(56)

L'encoder ammortizzato fattorizza:

$$q\theta \; (s1{:}T \; |x1{:}T \;) \; = \; \Pi t \; = \; 1 \; T \;\; q\theta \; (st \; |ht \; , xt \;) (57)$$

Sostituendo nell'ELBO e separando per passi temporali:

$$ELBO \; = \; \Sigma t \; = \; 1 \; T \;\; \mathbb{E}q \; [\ln p\theta \; (xt \; |st \;)] \;\; (58)$$

$$- \; \Sigma t \; = \; 1 \; T \;\; \mathbb{E}q \; [KL(\; q\theta \; (st \; |ht \; , xt \;) \; \parallel \; p\theta \; (st \; |ht \;) \;)] (59)$$

18.3 Forma chiusa della KL tra gaussiane diagonali

Dimostrazione — KL(𝒩(μ_q, σ_q²) ‖ 𝒩(0, I))

Per una gaussiana univariata $q = \mathcal{N}(\mu, \sigma^2)$ e $p = \mathcal{N}(0, 1)$:

$$KL(q\|p) \; = \; \int q(x) \, \ln \frac{q(x)}{p(x)} \, dx (60)$$

$$= \; \mathbb{E}q \; [- \frac{(x-\mu)^2}{2\sigma^2} \; - \; \ln \sigma \; + \; \frac{x^2}{2}] (61)$$

$$= \; - \frac{\sigma^2}{2\sigma^2} \; - \; \ln \sigma \; + \; \frac{\mu^2+\sigma^2}{2} (62)$$

usando $\mathbb{E}_q[x^2] = \mu^2 + \sigma^2$ e $\mathbb{E}_q[(x-\mu)^2] = \sigma^2$:

$$= \tfrac{1}{2}(\mu^2 + \sigma^2 - 1 - \ln \sigma^2) \quad (63)$$

Per d dimensioni indipendenti con prior $\mathcal{N}(0, I_d)$, la KL si somma per componenti:

$$KL(\mathcal{N}(\mu, diag(\sigma^2)) \,\|\, \mathcal{N}(0, I)) = \tfrac{1}{2} \Sigma i = 1\ d\ (\mu^2 i + \sigma^2 i - 1 - \ln \sigma^2 i) \quad (64)$$

□

18.4 KL tra distribuzioni categoriali

Per due categoriali $q = (q_1, \ldots, q_K)$ e $p = (p_1, \ldots, p_K)$, la KL è:

$$KL(q \,\|\, p) = \Sigma k = 1\ K\ qk\ \ln \frac{q\_k}{p\_k} \quad (65)$$

Per prior uniforme $p_k = 1/K$:

$$KL(q \,\|\, Unif) = \Sigma k\ qk\ (\ln qk + \ln K) = \ln K - H(q) \quad (66)$$

dove $H(q) = -\Sigma_k\ q_k\ ln\ q_k$ è l'entropia. Minimizzare la KL rispetto al prior uniforme equivale a massimizzare l'entropia del posterior — incentivando distribuzioni diffuse che non collassano su un singolo valore.

18.5 Il reparameterization trick: dimostrazione e confronto con REINFORCE

Il gradiente dell'ELBO rispetto a θ include il termine:

$$\nabla\theta\ \ \mathbb{E}q\ \theta\ [f(z)] = \nabla\theta \int q\theta\ (z|x)\ f(z)\ dz \quad (67)$$

Dimostrazione — Reparameterization trick vs REINFORCE

Il ***REINFORCE estimator*** (Williams 1992) porta il gradiente sotto il segno di integrale:

$$\nabla\theta \int q\, f\, dz = \int f\, \nabla\theta\ q\, dz = \int f \cdot q\, \nabla\theta\ ln\, q\, dz = \mathbb{E}q\ [f(z)\, \nabla\theta\ ln\, q(z|x)] \quad (68)$$

La terza uguaglianza usa $\nabla q = q\ \nabla\ ln\ q$. Questo estimatore è non distorto ma ad alta varianza, poiché $f(z)\ \nabla\ ln\ q$ può avere fluttuazioni enormi per diversi campioni di z.

Il ***reparameterization trick*** riscrive $z = g_\theta(\varepsilon)$, $\varepsilon \sim p(\varepsilon)$ (esempio: $z = \mu_\theta + \sigma_\theta \odot \varepsilon$, $\varepsilon \sim \mathcal{N}(0,I)$), allora:

$$\nabla\theta\ \ \mathbb{E}\varepsilon\sim p(\varepsilon)\ [f(g\theta\ (\varepsilon))] = \mathbb{E}\varepsilon\sim p(\varepsilon)\ [\nabla\theta\ \ f(g\theta\ (\varepsilon))] \quad (69)$$

La terza uguaglianza vale perché $p(\varepsilon)$ non dipende da θ, permettendo di portare ∇_θ dentro la media Monte Carlo. Il gradiente $\nabla_\theta f(g_\theta(\varepsilon)) = \partial f/\partial z \cdot \partial g/\partial\theta$ sfrutta tutta la struttura differenziabile di f — tipicamente il segnale di ricostruzione $\partial\ ln\ p(x|z)/\partial z$ — mentre REINFORCE usa solo la magnitudine scalare $f(z)$. La varianza del reparameterization è tipicamente 10–100× inferiore.

□

18.6 Straight-through: dimostrazione algebrica

Dimostrazione — Straight-through estimator

Sia $z = one\_hot(argmax(l))$ il campionamento discreto. Definiamo lo ***straight-through*** tramite la decomposizione:

$$z\_ST = one\_hot(k\ *) + sg(softmax(l)) - sg(one\_hot(k\ *)),\quad k\ * = argmax(l) \quad (70)$$

Forward pass: sg(·) azzera i suoi argomenti nel calcolo del valore, quindi $z_{ST} = one\_hot(k^*)$ — il corretto campione discreto.

Backward pass: il gradiente di z_{ST} rispetto a l è:

$$\partial z\_ST/\partial l = \partial one\_hot(k^*)/\partial l + \partial softmax(l)/\partial l - \partial sg(one\_hot(k^*))/\partial l \quad (71)$$

$$= 0 + J\_softmax(l) - 0 = J\_softmax(l) \quad (72)$$

La $\partial one\_hot(k^*)/\partial l = 0$ perché argmax è costante quasi ovunque, e $\partial sg(one\_hot(k^*))/\partial l = 0$ per definizione di stop-gradient. Il gradiente approssimato è quindi lo Jacobiano $J_{softmax}(l) = diag(s) - ss^T$ dove $s = softmax(l)$ — un'approssimazione biased ma a varianza nulla (deterministica), superiore a REINFORCE in pratica per reti profonde.

□

18.7 Operatore di Bellman e convergenza

Dimostrazione — Contrazione dell'operatore di Bellman

L'operatore di Bellman associato a una politica π è:

$$(\mathcal{T}\pi\ V)(s) = \mathbb{E}a\sim\pi\ [r(s,a)] + \gamma\ \mathbb{E}s'\sim p\ [V(s')] \quad (73)$$

Dimostriamo che $\mathcal{T}$ è una γ-contrazione sotto $\|\cdot\|_\infty$. Siano V, V' $\in \mathbb{R}^{|\mathcal{S}|}$. Allora:

$$|(\mathcal{T}\pi\ V)(s) - (\mathcal{T}\pi\ V')(s)| = \gamma\ |\mathbb{E}s'\sim p\ [V(s') - V'(s')]| \quad (74)$$

$$\leq \gamma\ \mathbb{E}s'\sim p\ [|V(s') - V'(s')|] \leq \gamma\ \|V - V'\|\infty \quad (75)$$

Prendendo il sup su s:

$$\|\mathcal{T}\pi\ V - \mathcal{T}\pi\ V'\|\infty \leq \gamma\ \|V - V'\|\infty \quad (76)$$

Poiché $\gamma < 1$, per il Teorema di Banach esiste un unico punto fisso V^{π} che soddisfa $V^{\pi} = \mathscr{T}^{\pi} V^{\pi}$. La sequenza $V_{k+1} = \mathscr{T}^{\pi} V_k$ converge a V^{π} con tasso γ^k:

$$\|Vk \quad - \quad V\pi \ \|\infty \ \leq \gamma k \ \ \|V0 \quad - \quad V\pi \ \|\infty \ (77)$$

□

In Dreamer, l'apprendimento del critico tramite i λ-return è un'istanza di questa convergenza iterativa: il target $V_{\lambda}(s_{\tau})$ approssima il punto fisso dell'operatore di Bellman, e la rete critica stima la distanza da questo punto fisso.

18.8 Bias-variance tradeoff dei λ-return: analisi

I λ-return (eq. 23) generalizzano il TD(λ) di Sutton. Per $\lambda \to 0$:

$$V\lambda \to 0 \ \ (s\tau \ ) \to r\tau \quad + \ \gamma \ v\psi \ \ (s\tau \ + \ 1 \ ) \ \ [bootstrap \ 1 \ - \ step] (78)$$

Questo è il TD(0) classico: bassa varianza (un solo campione di ricompensa), bias proporzionale all'errore del critico $|v_{\psi} - V^{\pi}|$. Per $\lambda \to 1$:

$$V\lambda \to 1 \ \ (s\tau \ ) \to \Sigma n \ = \ \tau \ t \ + \ H \ \ \gamma n \ - \ \tau \ \ rn \quad [Monte \ Carlo] (79)$$

La varianza del Monte Carlo cresce come $O(H \cdot \sigma^2_r)$ con l'orizzonte H. Per $\lambda = 0.95$ (default DreamerV3) e H = 16:

$$Var[V\lambda \ = \ 0.95 \ \ (s)] \approx (1 \ - \ \lambda) 2 \ \ \Sigma k \ = \ 0 \ H \ - \ 1 \ \ \lambda 2k \ \ Var[Vk \ N \ ]$$

(80)

La varianza totale è dominata dalle stime a pochi passi (k piccolo) grazie al decadimento geometrico λ^{2k}. Il bias residuo è $O(\lambda^H)$ — trascurabile per H = 16, $\lambda = 0.95$ ($0.95^{16} \approx 0.44$, che non è trascurabile; per questo DreamerV3 usa H=16 e non H=∞).

18.9 L'ELBO come Obiettivo di Controllo: Connessioni con il Free Energy Principle

Il framework variazionale dell'ELBO (sezione 18.1) stabilisce una connessione profonda con il Free Energy Principle di Friston (2010), che propone che il sistema nervoso biologico e, per estensione, qualsiasi agente cognitivo, minimizzi una quantità chiamata free energy variazionale F, strettamente correlata all'ELBO. La free energy variazionale si decompone come F = -ELBO = D_{KL}[q(z|o) ‖ p(z)] - E_{q}[log p(o|z)], dove q(z|o) è il posterior approssimato (il modello percettivo dell'agente) e p(z) è il prior (le aspettative generative dell'agente). Minimizzare F rispetto a q equivale a massimizzare l'ELBO (aggiornamento percettivo); minimizzare F rispetto alle azioni equivale a selezionare azioni che rendono le osservazioni future coerenti con le aspettative generative (controllo attivo). Questa dualità percez-azione caratterizza i modelli di active inference, che unificano percezione e controllo nello stesso framework variazionale.

$$F[q] = D_{KL}[q(z|o) \,\|\, p(z|a)] - E_{q(z|o)}[\log p(o|z)] \quad (18.9)$$

La connessione con il world model robotico è diretta: il prior p(z|a) nel Free Energy Principle corrisponde al modello di transizione T(z'|z,a) del world model (sezione 16.6); il posterior q(z|o) corrisponde all'encoder percettivo φ(o) = z; la minimizzazione di F rispetto alle azioni corrisponde alla pianificazione latente (sezione 27.5). I sistemi di active inference applicati alla robotica (Pezzato et al. 2020; Sancaktar et al. 2020) implementano questa struttura usando un world model gaussiano per T e un encoder lineare per φ, raggiungendo prestazioni comparabili a SAC su task di manipolazione semplici con il vantaggio di una giustificazione teorica unificata per percezione e controllo.

18.10 Convergenza delle λ-Return: Analisi della Varianza

Le λ-return $G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$ interpolano tra TD(0) (n=1, bassa varianza, alto bias) e Monte Carlo (λ=1, zero bias, alta varianza). L'analisi del bias-variance tradeoff (sezione 18.8) mostra che la varianza delle λ-return cresce come λ^{2n} con l'orizzonte: per $\lambda < 1$, la serie converge e la varianza è limitata. Formalmente, $\mathrm{Var}[G_t^\lambda] = (1-\lambda)^2 \sum_{n=1}^{\infty} \lambda^{2(n-1)} \mathrm{Var}[G_t^{(n)}]$. Se $\mathrm{Var}[G_t^{(n)}] = O(n)$, la varianza delle λ-return è $O((1-\lambda)^2 / (1-\lambda^2)^2) = O(1/(1+\lambda)^2)$, che decresce monotonicamente in λ. Il valore ottimale λ^* dipende dalla struttura della funzione di valore: se il valore è quasi-lineare nello stato, il bias di TD(0) è basso e λ^* è piccolo; se la funzione di valore ha forte non-linearità, λ^* deve essere più alto per ridurre il bias. In pratica, DreamerV3 (sezione 17.7) usa $\lambda = 0.95$, un valore empiricamente ottimale su un ampio range di task che bilancia bias e varianza nelle stime del valore per il planning nel world model.

Capitolo 19 — World Models per la Robotica Embodied

19.1 Il POMDP robotico: peculiarità strutturali

Il POMDP robotico si caratterizza per: spazio delle azioni continuo $\mathscr{A} \subseteq \mathbb{R}^{d_a}$ con d_a compreso tra 6 (braccio 6 DOF) e 35+ (umanoide); orizzonte per episodio tipicamente 500–2000 passi; ricompense sparse e spesso binarie (successo/fallimento); funzione di transizione che include dinamiche di contatto, deformazione e attrito, non esprimibili in forma analitica chiusa. Le osservazioni sono immagini RGB o depth map, con variazioni di illuminazione e occlusioni parziali.

Il costo per interazione fisica — tempo, usura hardware, rischio di danni — rende il paradigma world model particolarmente attraente: ridurre di un ordine di grandezza le interazioni fisiche necessarie ha implicazioni pratiche enormi. Con DreamerV3, un braccio robotico impara task di manipolazione visiva con ~500K passi reali invece dei ~5M richiesti da un algoritmo model-free come DrQ-v2.

19.2 Pianificazione latente come MPC neurale

Il world model funge da sostituto del simulatore fisico in un'architettura Model Predictive Control (MPC) neurale. Ad ogni passo di controllo t si ottimizza:

$$a * t{:}t + H \quad = \; argmaxa \; t{:}t + H \;\; \mathbb{E}p \; \phi \; [\Sigma\tau = t \;\; t + H \;\; \gamma\tau - t \;\; \hat{r}\tau \;\; | \, st \; , \; at{:}t + H \;]$$

(81)

Il MPC neurale evita la necessità di un modello analitico delle dinamiche fisiche, ma soffre del problema degli errori cumulativi sulle traiettorie

immaginate. Se l'errore medio di predizione a un passo è ε, l'errore cumulativo su H passi è limitato superiormente da (derivazione seguente):

$$errH \leq \varepsilon \cdot \Sigma k = 0\ H - 1\ (\gamma\ Lp\ )k = \varepsilon \cdot \frac{1 - (\gamma L\_p)\hat{}H}{1 - \gamma L\_p}\ (per\ \gamma L\_p \neq 1)$$

(82)

Dimostrazione — Bound sugli errori cumulativi

Sia $\tilde{s_t}$ lo stato reale e $\hat{s}_t$ lo stato simulato. Il prior dinamico ha costante di Lipschitz L_p:

$$\|p\phi\ (\cdot|\hat{s}t - 1\ , a) - p\phi\ (\cdot|\tilde{s}t - 1\ , a)\| \leq Lp\ \|\hat{s}t - 1 - \tilde{s}t - 1\ \|$$

(83)

L'errore si accumula ricorsivamente. Definendo $e_t = \|\hat{s}_t - \tilde{s_t}\|$, e usando la somma telescopica:

$$et \leq Lp\ et - 1 + \varepsilon \implies et \leq \varepsilon\ \Sigma k = 0\ t - 1\ Lk\ p$$ (84)

Il ritorno atteso cumulato è pesato da γ, quindi l'errore effettivo è:

$$errH = \Sigma t = 1\ H\ \gamma t - 1\ et \leq \varepsilon\ \Sigma t = 1\ H\ \gamma t - 1\ \Sigma k = 0\ t - 1\ Lk\ p = \varepsilon\ \Sigma k =$$

(85)

□

Per $\gamma\ L_p < 1$, l'errore rimane limitato per orizzonti arbitrari. Per $\gamma\ L_p \geq 1$, l'errore diverge esponenzialmente — manifestando il distributional shift nelle traiettorie immaginate.

19.3 Incertezza epistemica e aleatoria: ensemble methods

È fondamentale distinguere due sorgenti di incertezza nelle predizioni del world model. L'***incertezza aleatoria*** è intrinseca alla stocasticità delle dinamiche fisiche — persiste anche con dati infiniti. L'***incertezza epistemica*** dipende dalla mancanza di dati in regioni dello spazio degli stati non esplorate — decresce raccogliendo più esperienza. Confondere le due conduce a comportamenti eccessivamente cauti in regioni sicure ma non esplorate, e a comportamenti rischiosi in regioni note-pericolose.

Il metodo dell'ensemble di N world model $\{p_{\phi i}\}_{i=1}^{N}$ stima l'incertezza epistemica tramite la varianza del disaccordo:

$$Uepi\ (s,a)\ =\ \frac{1}{N}\Sigma i\ =\ 1\ N\ \ \|\mu i\ (s,a)\ -\ \bar{\mu}(s,a)\|2\ (86)$$

L'incertezza aleatoria è stimata come la varianza media dei modelli:

$$Ualea\ (s,a)\ =\ \frac{1}{N}\Sigma i\ \ Varp\ \phi\ i\ [s'\mid s,a](87)$$

Algoritmi come MBPO (Janner et al. 2019) limitano gli orizzonti di immaginazione alle regioni con U_{epi} < soglia, producendo traiettorie più affidabili. Il main trade-off è computazionale: N = 7 ensemble richiede 7× il costo di forward pass.

19.4 Domain randomization: formulazione matematica

Il domain randomization (Tobin et al. 2017, Andrychowicz et al. 2020) affronta il sim-to-real gap formalizzando il task di trasferimento come ottimizzazione su una distribuzione di MDP. Sia ξ un vettore di parametri fisici (attrito, massa, inerzia, parametri ottici) e sia $\mathcal{D}$ una distribuzione su ξ. L'obiettivo è:

$$max\pi\ \ \mathbb{E}\xi\sim\mathcal{D}\ [\mathbb{E}M\ \xi\ ,\pi[\Sigma t\ \ \gamma t\ \ rt\ ]](88)$$

La distribuzione $\mathscr{D}$ è progettata affinché il supporto includa l'ambiente reale ξ_{real} con probabilità non nulla. Se $\xi_{real} \in \text{supp}(\mathscr{D})$, allora la policy che massimizza l'obiettivo su tutti i domini del supporto performerà bene anche su ξ_{real}. La scelta di $\mathscr{D}$ è critica: un supporto troppo ristretto non copre il dominio reale; troppo ampio rende il problema sovra-vincolato.

19.5 Sim-and-Real Co-Training: Maddukuri et al. (2025)

13.5.1 Formulazione del problema

Il paper di Maddukuri, Jiang, Chen, Nasiriany et al. (UT Austin / NVIDIA 2025) studia sistematicamente il ***co-training su dati misti***: anziché trasferire una policy addestrata interamente in simulazione, si addestra la policy su un dataset misto $D = D_{real} \cup D_{sim}$. La funzione di perdita di behavioral cloning sul dataset misto è:

$$\mathcal{L}total\ (\theta;\ Dreal\ ,\ Dsim\ ) = \alpha \cdot \mathcal{L}(\theta;\ Dsim\ ) + (1 - \alpha) \cdot \mathcal{L}(\theta;\ Dreal\ ) \quad (89)$$

$$dove\ \mathcal{L}(\theta;\ D) = \frac{1}{|D|}\Sigma(o,a)\in D \quad - log\ \pi\theta\ (a|o) \quad (90)$$

Il parametro $\alpha \in [0,1]$ è il ***co-training ratio***: la probabilità che ogni mini-batch campioni da D_{sim} anziché da D_{real}. La scelta di α è critica per le prestazioni. Con $\alpha = 0$ si ha solo dati reali; con $\alpha \to 1$ si ha quasi solo dati simulati (che non corrispondono al dominio reale).

13.5.2 Risultati sperimentali quantitativi

Lo studio si conduce su due embodiment: ***Panda Kitchen*** (Franka Emika Panda 7-DOF, setup DROID) con 3 task (CounterToSinkPnP, CounterToCabPnP, CloseDoor, 50 demo reali ciascuno) e ***Humanoid Tabletop*** (Fourier GR-1, 3 task: CupPnP, MilkPnP, Pouring, 20 demo reali ciascuno). Le sorgenti di dati simulati sono:

Dati task-agnostici (Prior): il dataset RoboCasa con 72k dimostrazioni su 24 task in 100 scene diverse, generato da 50 demo umane tramite MimicGen (3000 traiettorie per task). I dati Prior non sono allineati al dominio reale in termini di posizione del robot, parametri fisici e categorie di oggetti.

Digital cousin task-aware (DC): ambienti simulati creati appositamente per rispettare quattro vincoli: stesso robot e action space; stesso task goal e success check; stesse categorie di oggetti (ma istanze diverse per geometria e texture); stesse categorie di fixture ambientali. Le DC vengono popolate con 10.000 dimostrazioni per task generate da 100 demo umane con MimicGen.

I risultati principali (Tabella I del paper) mostrano:

Real only: 45.3% di success rate medio su 6 task.

Real + DC: 81.1% (+35.8pp). Il co-training con DC è il caso più allineato: stesse categorie di oggetti, stessa geometria del workspace. Il miglioramento più drammatico è su CloseDoor: da 10% a 100%.

Real + Prior: 76.8% (+30.5pp). Risultato sorprendente: anche dati simulati senza alcuna conoscenza a priori del task reale migliorano sostanzialmente le prestazioni.

Real + DC + Prior: 83.2% (+37.9pp). La combinazione di tutte le sorgenti produce il risultato migliore.

13.5.3 Analisi del co-training ratio α

L'effetto di α è non monotonico. Per il task CupPnP con 20 demo reali e 1000 demo simulate, il success rate varia da 0.17 ($\alpha = 0.1$) a 0.95 ($\alpha = 0.90$) a 0.80 ($\alpha = 0.99$). La regione ottimale è $\alpha \in [0.7, 0.99]$. Un α eccessivamente basso non sfrutta i dati simulati; uno eccessivamente alto fa sì che il

modello apprenda prevalentemente dal dominio simulato, peggiorando il trasferimento.

La loss totale (eq. 89) mostra che α deve essere scelto in modo che il gradiente del dominio simulato non sopraffaccia quello del dominio reale. Una regola empirica derivata dallo studio è:

$$\alpha^* \approx 1 - \frac{|D\_real|}{|D\_real| + |D\_sim|} \approx 0.95 \quad (per\ |D\_sim| \gg |D\_real|) \quad (91)$$

13.5.4 Generalizzazione a oggetti e posizioni non viste

Il co-training con DC migliora anche la generalizzazione out-of-distribution (Tabella II del paper):

Oggetti non visti (CounterToSinkPnP, 8 nuove categorie): 33% (solo Real) → 50% (Real+DC) sul Panda; 10% → 80% sull'umanoide.

Posizioni non viste (oggetti nella zona centrale, non presenti nelle demo reali): 11% → 28% sul Panda; 43% → 100% sull'umanoide.

L'interpretazione teorica è che la simulazione agisce come un ***curriculum implicito***: espone la policy a una distribuzione più ampia di stati iniziali e configurazioni di oggetti, inducendo una rappresentazione latente più invariante alle trasformazioni del dominio.

13.5.5 Co-training in regime data-rich

Il beneficio del co-training persiste anche con dataset reali più grandi (MultiTaskPnP, Humanoid, 40–400 demo reali, 4000 DC simulate). Con 400 demo reali, Real only ottiene 0.39 e Real+DC ottiene 0.78 (+100%). Il co-training rimane vantaggioso anche in regime data-rich, suggerendo che il beneficio non è solo quantitativo (più dati) ma qualitativo (diversità delle traiettorie simulate rispetto alle dimostrazione umane tipicamente correlate).

19.6 Genie: foundation world model non supervisionato

Genie (Bruce, Dennis, Edwards, Parker-Holder et al., Google DeepMind 2024) è un foundation world model da 11 miliardi di parametri addestrato su oltre 200.000 ore di video di gameplay non etichettati. Apprende uno spazio di azioni latenti in modo completamente non supervisionato.

13.6.1 Latent Action Model (LAM)

Il LAM inferisce le azioni latenti tra frame consecutivi senza etichette. L'encoder prende come input tutti i frame precedenti $x_{1:t}$ e il frame successivo x_{t+1} e produce un'azione latente continua $\tilde{a}_t$. Un decoder ausiliario ricostruisce il prossimo frame da $(x_{1:t}, \tilde{a}_t)$. Per limitare il vocabolario delle azioni, la ã viene quantizzata tramite VQ con |A| = 8 codici:

$$at = argmini\in A \ \|\tilde{a}t - ei\|2 , \quad ei \in \mathbb{R}d \quad (92)$$

13.6.2 Video Tokenizer (ST-ViViT)

Il tokenizer VQ-VAE basato su ST-Transformer (Spatiotemporal ViT) comprime ogni frame in D token discreti. L'ST-Transformer interleave attenzione spaziale (sui H×W token di un singolo frame) e attenzione temporale causale (sullo stesso token attraverso i T frame). La complessità computazionale è O(T × H × W) per la componente spaziale dominante — lineare in T anziché quadratica come per l'attenzione piena.

13.6.3 Dynamics Model (MaskGIT Transformer)

Il dynamics model è un Transformer MaskGIT che predice i token del prossimo frame dati i token passati e l'azione latente corrente. È addestrato con cross-entropia su token mascherati con Bernoulli(p), p ~ Uniforme[0.5, 1.0]:

$$\mathcal{L}dyn \;=\; \mathbb{E}M{\sim}Bern(p)\;[-\;ln\,pG\;(\tilde{z}2{:}T\;\;|\,z1{:}T\;-\;1\;\;\odot(1\;-\;M),\;\tilde{a}1{:}T\;-\;1\;)] \quad (93)$$

L'analisi di scaling dimostra una legge di potenza del training loss in funzione dei parametri: da 40M a 2.7B parametri il loss segue $L \propto N^{-\alpha}$ con $\alpha \approx 0.07$ — un tasso di scaling più lento rispetto ai modelli linguistici ($\alpha \approx$ 0.08–0.10) ma qualitativamente simile, suggerendo che i world model beneficino delle stesse proprietà di scaling.

19.7 Curiosità intrinseca basata sul world model

In ambienti a ricompense sparse, il world model fornisce un segnale di esplorazione naturale: la ***sorpresa del modello***. Il guadagno di informazione atteso per l'azione (s,a) è:

$$rint\;t \;\;=\;\; KL[\,p(\theta\,|\,x1{:}t\;,\;a1{:}t\;)\,\|\,p(\theta\,|\,x1{:}t\;-\;1\;,\;a1{:}t\;-\;1\;)\,] \quad (94)$$

In pratica, si stima questa quantità tramite il ***prediction error*** del world model:

$$rint\;t\;\approx\|p\phi\;(\hat{s}t\;\;|\,st\;-\;1\;,\;at\;-\;1\;)\;-\;st\;\|2 \quad (95)$$

I metodi basati su curiosità intrinseca (ICM, Pathak et al. 2017; RND, Burda et al. 2019) hanno dimostrato benefici significativi su task con ricompense sparse come Montezuma's Revenge (Atari) e MiniGrid. L'integrazione con world model latenti è naturale: il prediction error sullo spazio latente $\|\hat{z}_t - z_t\|$ è computazionalmente molto più efficiente del prediction error sullo spazio delle immagini.

19.8 Trasferimento cross-embodiment

Una frontiera aperta della ricerca è il trasferimento di world model tra morfologie robotiche diverse. Un modello addestrato su un braccio 7-DOF non si trasferisce direttamente a un robot umanoide 35-DOF, poiché la struttura dello spazio delle azioni è radicalmente diversa.

Un approccio emergente è la ***rappresentazione dell'intenzione motoria***: mappare le azioni di diversi robot su uno spazio latente comune indipendente dalla morfologia. Formalmente, si apprende una funzione $\varphi: \mathcal{A}_{\text{robot}} \rightarrow \mathcal{Z}_{\text{intent}}$ che proietta le azioni dell'end-effector (velocity in Cartesian space) su un embedding comune. Il world model condiziona sulle intenzioni z_{intent} anziché sulle azioni giunti raw:

$$st + 1 \sim p\phi\ (st + 1 \mid st\ ,\ zintent, t\ ), \quad zintent, t\ = \varphi(at\ ) \quad (96)$$

Questa factorizzazione consente al world model di apprendere rappresentazioni invarianti alla morfologia per gli aspetti semantici del task (avvicinarsi, afferrare, posizionare) pur mantenendo la specificità cinematica per il calcolo dell'inverse kinematics necessario all'esecuzione fisica.

19.9 Sfide aperte: scalabilità, sicurezza e architetture ibride

La prima sfida è la ***manipolazione dexterus a lungo orizzonte***: task che richiedono migliaia di passi con dipendenze temporali a lungo raggio (assemblaggio meccanico, cucito, operazioni chirurgiche assistite). L'orizzonte di immaginazione H = 16 di DreamerV3 è inadeguato; estenderlo amplifica gli errori cumulativi (eq. 82). Soluzioni candidate: world model gerarchici con modelli a diversi livelli di astrazione temporale (abstract dynamics model + primitive dynamics model), o architetture di tipo episodic memory che consentono al Transformer di accedere a stati latenti molto lontani nella sequenza temporale.

La seconda sfida è la ***safety certificabile***: garantire che la policy appresa via world model non esegua azioni fisicamente pericolose quando il modello produce traiettorie in regioni out-of-distribution. Il ***conformal prediction*** offre garanzie probabilistiche: dato un insieme di calibrazione di traiettorie, si stima un quantile di coverage $1 - \delta$ tale che la policy sia sicura con probabilità $1 - \delta$ sul prossimo episodio. Integrato con l'incertezza epistemica del world model (eq. 86–87), questo approccio promette certificazioni soft a basso costo computazionale.

La terza sfida è l'***architettura ibrida simulazione-world model***: combinare un simulatore fisico differenziabile (e.g. MuJoCo, Isaac Gym) con un world model neurale per i residui non modellati (attrito adesivo, dinamiche tendinee, comportamento degli oggetti deformabili). Questo approccio ***residual dynamics model*** addestra il world model a predire il residuo:

$$st + 1 \;\; = \;\; fsim\;(st\;,\;at\;) \;\; + \;\; \Delta\phi\;\;(st\;,\;at\;) \quad (97)$$

dove f_{sim} è il simulatore deterministico e Δ_{ϕ} è il residuo neurale. Questa decomposizione migliora l'efficienza campionaria (il simulatore fornisce un strong prior) e riduce gli errori cumulativi (il residuo ha magnitudine piccola).

La quarta sfida, emersa chiaramente dal lavoro Genie, è il ***pre-training da video su larga scala***: addestrare un foundation world model su corpora eterogenei di video robotici (Open-X Embodiment, DROID, RT-1/RT-2) e successivamente fare fine-tuning su task specifici con poche dimostrazioni reali. Questa traiettoria, che rispecchia fedelmente quella dei grandi modelli linguistici (pre-training → fine-tuning → RLHF), è oggi la frontiera più promettente per la creazione di agenti robotici generalisti. I risultati del sim-and-real co-training (§13.5) forniscono evidenze che la diversità dei dati di pre-training — anche con domain gap significativo — si traduce in miglioramenti robusti sul task target.

PARTE IV

JEPA e la Visione di LeCun

Capitolo 20 — Energia e architetture EBM: la Joint Embedding Predictive Architecture

20.1 Il paradigma degli Energy-Based Models: motivazione epistemica

La modellazione statistica classica presuppone che ogni sistema di apprendimento possa essere ricondotto a una distribuzione di probabilità normalizzata. Addestrare Il sistema equivale a trovare i parametri che massimizzano la log-verosimiglianza dei dati, ovvero che minimizzano la KL divergenza tra la distribuzione empirica e il modello.Questa impostazione porta con sé un requisito imprescindibile: la funzione di partizione $Z(w) = \int \exp(-F_w(y))\, dy$ deve esistere e, idealmente, essere calcolabile. In spazi adalta dimensionalità — come quello dei pixel di un'immagine ad alta risoluzione, quello delle rappresentazioni intermedie di un transformer, o quello degli stati propriocettivi di un robot che manipola oggetti complessi — questo integrale è genericamente intrattabile.Le tecniche di approssimazione disponibili — campionamento MCMC, variational inference,contrastive divergence — introducono bias sistematici o costi computazionali che crescono esponenzialmente con la dimensionalità.

Yann LeCun ha argomentato sistematicamente, in una serie di lavori che culmina nel documento "A Path Towards Autonomous Machine Intelligence" (2022), che il vincolo della normalizzazione sia non necessario per la grande maggioranza delle applicazioni di apprendimento automatico. Se l'obiettivo è discriminare tra configurazioni compatibili e incompatibili,predire la rappresentazione di un segnale futuro, o pianificare sequenze di azioni che minimizzino un costo intrinseco, non serve alcuna distribuzione normalizzata. Ciò che

serve è una funzione scalare capace di ordinare le configurazioni: bassa per quelle coerenti con i dati, alta per quelle incoerenti. Questa funzione è l'energia, e i modelli che la adottano come oggetto fondamentale sono gli Energy-Based Models (EBM).

Le radici concettuali degli EBM affondano nella fisica statistica del XIX secolo.L'energia di Boltzmann, la funzione di Hamiltoniano nei modelli di Ising, la free energy di Gibbs sono funzioni scalari che descrivono la compatibilità di una configurazione con un insieme di vincoli fisici. Nella traduzione al machine learning, la funzione di energia F_w(x, y) — con parametri w, ingresso x e output y — prende il posto dell'Hamiltoniano:produce un numero reale basso quando x e y sono mutuamente coerenti, e un numero elevato quando non lo sono. A differenza dell'Hamiltoniano fisico, F_w non deve soddisfare alcun vincolo fisico: può essere qualunque funzione differenziabile parametrica, tipicamente implementata da una rete neurale profonda. Questa libertà è il principale vantaggio del formalismo EBM rispetto ai modelli probabilistici: l'intera classe delle reti neurali profonde diventa disponibile come spazio di ipotesi per F_w, senza vincoli di normalizzabilità.

20.2 Formalismo matematico degli EBM

Definizione formale. Un Energy-Based Model è una coppia (F_w, I) doveF_w : X × Y → $\mathbb{R}$ è una funzione scalare parametrica differenziabile e I è la procedura di inferenza che, dato x ∈ X, produce la stima:

$$\bar{y} = argmin_{y \in Y} F_w(x, y) \quad (14.1)$$

La connessione con la probabilità è opzionale. Se si desidera una densità condizionale,si può definire la distribuzione di Gibbs a temperatura inversa β:

$$p_w(y \mid x) = \frac{exp(-\beta F_w(x,y))}{Z_w(x, \beta)} \quad (14.2)$$

dove la funzione di partizione Z_w(x, β) = ∫◎ exp(−β F_w(x,y)) dy. Questo collegamento è una scelta e non un obbligo strutturale: l'EBM può essere definito, addestrato e utilizzato senza mai fare riferimento a Z_w. Nel limite β → ∞ la distribuzione di Gibbs concentra tutta la massa attorno al minimo di F_w rispetto a y: l'inferenza a temperatura zero coincide con l'equazione (14.1). Questo limite è quello operativamente rilevante per JEPA.

20.3 Latent-Variable EBM: formalismo e dimostrazione della free energy

In molti scenari pratici la compatibilità tra x e y dipende da informazione non contenuta in x e non direttamente osservabile in y, ma che costituisce un grado di libertà interno al modello. Questa informazione è la variabile latente z. Un Latent-Variable EBM (LVEBM) è una funzione E_w(x, y, z) di tre argomenti. L'inferenza comprende due fasi:

$$\dot{z} \;=\; argmin_{z\in Z}\, E_{\;w}(x,\, y,\, z) \quad (14.3)$$

poi si elimina z per ottimizzazione, ottenendo la free energy a temperatura zero:

$$F_{\;w}(x,\, y) \;=\; min_{z\in Z}\, E_{\;w}(x,\, y,\, z) \;=\; E_{\;w}(x,\, y,\, \dot{z}) \quad (14.4)$$

Dimostrazione: la free energy come limite a T→0 della free energy di Gibbs. Nella formulazione probabilistica, la free energy di Gibbs della variabile latente èF(x, y, β) = −(1/β) log Z_w(x, y, β), dove Z_w(x, y, β) = ∫ exp(−β E_w(x, y, z)) dz.Espandendo con il metodo di Laplace attorno al punto di minimo ż = argmin_z E_w:

$$\int exp(-\,\beta\, E),\; dz \approx exp(-\,\beta\, E(\dot{z})) \cdot (2\pi/\beta)^{\;d\_z/2} \cdot det(H)^{\;-1/2} \quad (14.5)$$

dove H = ∂²E_w/∂z² valutata in ż è la hessiana. Sostituendo nella free energy:

$$F(x, y, \beta) \approx E_{w}(x, y, \dot{z}) + \frac{d\_z}{2} log(2\pi/\beta) + \frac{1}{2\beta} log\ det(H) \quad (14.6)$$

Quando β → ∞, i termini O(log β) tendono a −∞ rispetto a E_w(x,y,ż) che è O(1):F(x,y,β) → E_w(x,y,ż) = $min_z E\_w(x, y, z)$. Questo giustifica l'uso del termine

free energy a temperatura zero per l'equazione (14.4): è il limite termodinamico di una quantità probabilistica ben definita, non un artificio computazionale.La hessiana H deve essere definita positiva: se ż è un punto di sella anziché un minimo,il metodo di Laplace è inapplicabile e la free energy deve essere calcolata con metodi più sofisticati. In JEPA, la regolarizzazione di z (discussa nella Sezione 15.5) garantisce che il problema di minimizzazione (14.3) sia ben posto e ammetta un minimo unico o un insieme compatto di minimi.

20.4 Addestramento degli EBM: collasso e metodi di contrasto

Definizione formale di collasso. Un EBM con parametri w* collassa seF_w*(x, y) ≈ c per quasi ogni (x, y) ∈ X × Y. Il sistema perde ogni utilità discriminativa:attribuisce energia identica a punti sul manifold dei dati e a punti lontani da esso.Il collasso è la soluzione banale di qualunque procedura di addestramento che minimizzi F_w sui campioni senza un contrappeso che impedisca l'appiattimento dell'intera superficie.

Metodi contrastivi. Un metodo contrastivo costruisce campioni negativi ȳ esplicitamente fuori dal manifold e impone F_w(x, ȳ) > F_w(x, y) + m per un margine m > 0. La forma generale della perdita contrastiva è:

$$L_c(w) = E_{(x,y)\sim p}[F_w(x,y)] - E_{(x,\bar{y})\sim q}[F_w(x,\bar{y})] \quad (14.7)$$

La hinge loss con margine dipendente dalla distanza è un caso particolare:L = [F_w(x,y) − F_w(x,ȳ) + μ‖y − ȳ‖²]₊, dove [·]₊ = max(0,·).

InfoNCE e bound sull'informazione mutua. La perdita InfoNCE, utilizzata daSimCLR, MoCo e CLIP, con K campioni negativi ȳ[1],...,ȳ[K] estratti da q(y) è:

$$L_{NCE} = -E\left[\log \frac{exp(-F_w(x,y))}{exp(-F_w(x,y)) + \Sigma^k exp(-F_w(x,\bar{y}[k]))}\right] \quad (14.8)$$

Teorema (bound InfoNCE). Per ogni distribuzione di campionamento q, ogni K ≥ 1,e ogni funzione energia F_w vale:

$$I(x;\ y) \geq \log(K) - L_{NCE} \quad (14.9)$$

Dimostrazione. Sia r_w(x,y) = exp(−F_w(x,y)). Scriviamo la perdita NCE come l'attesa negativa di un rapporto di verosimiglianza normalizzato. Per la disuguaglianza di Jensen applicata alla funzione concava log(·):

$$E_p[\log r(x,y)] \geq E_p\left[\log \frac{r(x,y)/K}{(1/K)r(x,y) + (1/K^2)\Sigma_k r(x,\bar{y}[k])}\right] = \log K - L_{NCE} \quad (14.10)$$

Poiché I(x;y) = KL(p(x,y) ‖ p(x)p(y)) = E_p[log(p(y|x)/p(y))] = E_p[log r(x,y)] − const,il bound (14.9) segue. Il bound è stretto quando K → ∞ e q = p(y) (distribuzione marginale di y). Nella pratica, K è la dimensione del batch meno 1: per K = 2048(tipico in CLIP) il bound vale log(2048) ≈ 7.6 bit, sufficiente per applicazioni pratiche.

Maledizione della dimensionalità nei metodi contrastivi. Per coprire uniformemente la sfera S^{D-1} in ℝ^D con K campioni a distanza massima ε l'uno dall'altro,servono K = O((1/ε)^{D-1}) campioni. In D = 1280 (ViT-Huge) questo esponente rende il numero di campioni

astronomico anche per $\varepsilon = 0.1$. Questo è il motivo strutturale per cui i metodi contrastivi sono scalabili solo fino a certe dimensionalità e richiedono batch di grandi dimensioni per funzionare bene.

Metodi regolarizzati. Evitano il problema costruendo un termine R(w) che penalizza il volume della regione a bassa energia, senza campioni negativi:L_reg(w) = E_{(x,y)~p}[F_w(x,y)] + λ R(w). La scelta di R dipende dall'architettura.Per JEPA, R include la regolarizzazione di varianza e covarianza degli embedding (VICReg)o la penalizzazione dei termini fuori diagonale della matrice di cross-correlazione (Barlow Twins).

20.5 Architetture standard e fenomenologia del collasso

Architettura predittiva deterministica. Produce $\bar{y} = f(x)$ e l'energia è $D(y, \bar{y})$. Non può collassare perché per ogni x c'è un unico $\bar{y}$ con energia zero. Ma non può modellare distribuzioni $p(y|x)$ multimodali: quando $p(y|x)$ è bimodale con mode y_1e y_2, il predittore converge su $(y_1+y_2)/2$, la media delle mode, che non corrisponde ad alcun punto reale della distribuzione. In robotica, questo è un limite grave: la stima della posizione futura di un oggetto ambiguo (ad esempio un disco che può cadere a sinistra o a destra) converge sulla media delle due traiettorie, producendo una previsione che non corrisponde a nessuna fisica reale.

Architettura generativa a variabile latente. Il decoder produce $\bar{y}$ = Dec(f(x), z).Può modellare distribuzioni multimodali se z ha sufficiente capacità. Ma può collassare se dim(z) ≥ dim(y): il decoder apprende la funzione Dec(f(x), z) = z, ignorando f(x).La perdita D(y, Dec(f(x), z)) si annulla con z = y, collassando l'energia su tutto Y.

Auto-encoder. Se dim(s) ≥ dim(y), può apprendere la funzione identità:D(y, Dec(Enc(y))) = 0 per ogni y. Collassa nella soluzione triviale di ricostruzione perfetta senza rappresentazione semantica.

Joint Embedding Architecture (JEA). Produce s_x = Enc_x(x), s_y = Enc_y(y)e misura D(s_x, s_y). Collassa se gli encoder imparano s_x = s_y = c (costante): l'energia è D(c, c) = 0 su tutto X × Y.

Joint Embedding Predictive Architecture (JEPA). Produce s_x = Enc_x(x),s_y = Enc_y(y), e la previsione $\tilde{s}$_y = Pred(s_x, z). L'energia è D(s_y, $\tilde{s}$_y). Il collasso è prevenuto più efficacemente tramite: (a) architettura asimmetrica con target encoderEMA; (b) regolarizzazione non contrastiva di s_x e s_y tramite VICReg o Barlow Twins;(c) regolarizzazione di z. La predizione avviene nello spazio compresso delle rappresentazioni,eliminando la necessità di predire dettagli irrilevanti come la texture a livello di pixel.

20.6 Dalla fisica all'intelligenza: il collasso come problema cognitivo

Il formalismo EBM non è solo un artificio computazionale: riflette una visione sulla natura della cognizione. Gli esseri viventi non operano costruendo distribuzioni di probabilità sull'intero spazio degli eventi possibili. Valutano la compatibilità di situazioni percepite con modelli interni del mondo, identificando stati a bassa energia cognitiva (coerenti,plausibili, normali) e stati ad alta energia (incoerenti, sorprendenti, pericolosi). La predictive coding, proposta da Rao e Ballard (1999) come modello della corteccia visiva,è formalmente un EBM gerarchico: ogni livello della gerarchia corticale predice le attivazioni del livello inferiore, e gli errori di predizione — la differenza tra previsione e realtà — vengono propagati verso l'alto come segnali di aggiornamento. JEPA è la realizzazione computazionale moderna di questa visione.

Per la robotica, il formalismo EBM apre prospettive concrete. Un robot che apprende a manipolare oggetti in un ambiente fisico ha bisogno di un modello interno del mondo che assegni bassa energia alle

configurazioni di oggetti fisicamente stabili e agli stati del braccio meccanico coerenti con le leggi della cinematica, e alta energia alle configurazioni instabili o cinematicamente impossibili. JEPA, addestrato su dati visivi e propriocettivi del robot in interazione con l'ambiente, può costruire automaticamente tale superficie energetica senza supervisione esplicita. Il modulo Critico dell'architettura cognitiva di LeCun svolge esattamente questa funzione: predice il valore futuro del costo intrinseco, guidando il pianificatore verso sequenze di azioni che portano il robot verso stati a bassa energia (compito completato, oggetto afferrato in modo stabile,postura sicura).

Capitolo 21 — La matematica di JEPA: spazi di embedding, funzioni di costo, VICReg e Barlow Twins

21.1 Definizione formale e struttura geometrica

Siano X, Y spazi di dati (tipicamente ℝ^{d_X} e ℝ^{d_Y}) e Z lo spazio della variabile latente. Un JEPA è definito da una tripla di funzioni differenziabili: Enc_x : X → ℝ^d, Enc_y : Y → ℝ^d, Pred : ℝ^d × Z → ℝ^d.Le rappresentazioni sono s_x = Enc_x(x) e s_y = Enc_y(y), entrambe in ℝ^d.La previsione è s̃_y = Pred(s_x, z). L'energia del sistema per la terna (x, y, z) è:

$$E_w(x, y, z) = D(Enc_y(y), Pred(Enc_x(x), z)) \quad (15.1)$$

Dopo la marginalizzazione su z (equazione 14.4), la free energy è:

$$F_w(x, y) = min_{z \in Z} D(s_y, Pred(s_x, z)) \quad (15.2)$$

Geometria dello spazio di embedding. Per un dato s_x, l'immagine del predittore è la varietà Ψ(s_x) = {Pred(s_x, z) : z ∈ Z} ⊂ ℝ^d. Se dim(Z) = d_z ≤ d ePred è una funzione differenziabile, Ψ(s_x) ha dimensione al più d_z. L'immagine del target encoder è Φ = {Enc_y(y) : y ∈ Y} ⊂ ℝ^d: il manifold delle rappresentazioni target. Un JEPA ben addestrato porta Ψ(s_x) vicino alla proiezione su Φ dei punti y compatibili con x, senza dover modellare l'intera struttura di Y. Il rapporto di riduzione dimensionale r = dim(Y)/d quantifica il guadagno computazionale: per un'immagine 224×224×3 e un ViT-Huge (d = 1280), r ≈ 118.

21.2 I quattro criteri di addestramento non contrastivo: necessità e sufficienza

LeCun identifica quattro condizioni necessarie per l'addestramento non contrastivo di un JEPA. La necessità di ciascuna può essere dimostrata mostrando un controesempio in cui le altre tre sono soddisfatte ma la quarta no, e il JEPA collassa.

Criterio I: max I(s_x; x). **Criterio II:** max I(s_y; y). **Criterio III:** min E[D(s_y, Pred(s_x, z*))]. **Criterio IV:** min I(z; y|x).

Dimostrazione della necessità del Criterio IV. Assumiamo i Criteri I–III soddisfatti e dim(Z) = d, con il predittore che implementa la funzione identità Pred(s_x, z) = z per ogni z. Per qualsiasi target s_y, il valore z* = s_y annulla la perdita: D(s_y, Pred(s_x, z*)) = D(s_y, s_y) = 0. La superficie energetica F_w(x,y) = 0 per ogni (x,y). Il Criterio I è reso inutile perché il predittore non usa mai s_x.Per prevenire questo collasso, l'informazione trasportata da z deve essere limitata. ■

Dimostrazione della necessità del Criterio II. Assumiamo i Criteri I, III, IV soddisfatti e Enc_y costante: Enc_y(y) = c per ogni y. Allora s_y = c, e il predittore apprende Pred(s_x, z) = c per ogni (s_x, z): D(c, c) = 0. La perdita è identicamente zero, ma la rappresentazione c non contiene informazione alcuna su y. Il Criterio II impedisce questa degenerazione imponendo che la distribuzione di s_y sul dataset abbia varianza positiva e componenti decorrelate. ■

21.3 VICReg: derivazione analitica completa con dimostrazioni

VICReg (Bardes, Ponce, LeCun — ICLR 2022) implementa i Criteri I e II tramite tre termini differenziabili applicati agli embedding z =

h_φ(f_θ(x)) e z' = h_φ(f_θ(x'))prodotti da un batch di n immagini con due viste aumentate.

15.3.1 Termine di invarianza

$$s(Z,\ Z') \ = \ \frac{1}{n}\Sigma_{i=1}^{n} \|z_{\ i} \ - \ z'_{\ i}\|_{\ 2}^{\ 2} \quad (15.3)$$

Minimizzare s spinge le rappresentazioni delle due viste della stessa immagine a coincidere. La soluzione banale z_i = z'_i = 0 annulla s: per questo motivo s da solo non è sufficiente e richiede i termini v e c.

15.3.2 Termine di varianza: dimostrazione della necessità della deviazione standard

Sia z^j ∈ ℝ^n il vettore delle j-esime componenti di tutti gli n embedding nel batch.La deviazione standard regolarizzata e il termine di varianza sono:

$$S(z^j,\ \varepsilon) \ = \ \sqrt{Var(z^j) \ + \ \varepsilon} \quad (15.4)$$

$$v(Z) \ = \ \frac{1}{d}\Sigma_{j=1}^{d} max(0,\ \gamma \ - \ S(z^j,\ \varepsilon)) \quad (15.5)$$

Dimostrazione della necessità della radice quadrata. Il gradiente di v rispettoalla componente z^j_i è:

$$\frac{\partial v}{\partial z^j_{\ i}} \ = \ -\frac{1}{d} \cdot \mathbb{1}[S \ < \ \gamma] \cdot \frac{z^j_{\ i} - \bar{z}^j}{(n-1) \cdot S(z^j, \varepsilon)} \quad (15.6)$$

Se si usasse la varianza V(x) = Var(x) invece di S(x) = √(Var(x)+ε):∂v/∂z^j_i ∝ (z^j_i − z̄^j). Nel limite di collasso, tutti gli embedding si avvicinano alla media: (z^j_i − z̄^j) → 0. Il gradiente è esattamente zero quando il collasso sta avvenendo — esattamente quando dovrebbe essere massimo. Con la deviazione standard,il denominatore S(z^j, ε) → √ε nel limite di collasso, e il gradiente diventa (z^j_i − z̄^j)/√ε:

rimane finito e non nullo (tranne nel caso degenere z^j_i = z̄^j per ogni i),garantendo un segnale di addestramento robusto. ■

15.3.3 Termine di covarianza: decorrelazione e codice fattoriale

La matrice di covarianza empirica di Z, centrata lungo il batch, e il relativo termine di perdita sono:

$$C(Z) = \frac{1}{n-1}\Sigma_{i=1}^{n}(z_i - \bar{z})(z_i - \bar{z})^{\top} \quad (15.7)$$

$$c(Z) = \frac{1}{d}\Sigma_{i\neq j}[C(Z)]_{ij}^{2} \quad (15.8)$$

Dimostrazione dell'efficacia di c contro il collasso informativo. Supponiamo Che le n rappresentazioni si concentrino su un sottospazio lineare V di dimensione k < d.La matrice C(Z) ha rango k, con autovalori λ_{k+1} = ... = λ_d = 0. Se le direzioni principali di C non coincidono con i versori canonici e_1,...,e_d, i termini fuori diagonale [C]_{ij} con i≠j sono non nulli, generando un gradiente che ruota le direzioni principali verso i versori canonici. Quando le direzioni principali coincidono con e_1,...,e_k,gli elementi fuori diagonale sono zero — ma ora k componenti hanno varianza zero.Il termine v interviene: impone che tutte le d dimensioni abbiano varianza ≥ γ, forzandole dimensioni k+1,...,d a diventare attive e portando d direzioni principali distinte a coprire lo spazio ℝ^d. La combinazione v + c è quindi necessaria: v da sola non decorrellerebbe, c da sola non eviterebbe il collasso di norma. ■

15.3.4 La funzione di perdita complessiva e l'espansore

$$\ell(Z, Z') = \lambda \cdot s(Z, Z') + \mu \cdot [v(Z) + v(Z')] + \nu \cdot [c(Z) + c(Z')] \quad (15.9)$$

L'obiettivo globale su tutto il dataset D e su tutte le coppie di trasformazioni t,t'∽Tè L = Σ_{I∈D} Σ_{t,t'∽T} ℓ(Z^I, Z'I). I valori ottimali secondo Bardes et al. sonoλ = μ = 25, ν = 1, ε = 10^{-4}, γ = 1. L'espansore h_φ mappa y da 2048 a 8192 dimensioni con una MLP a 3 strati. Il ruolo

dell'espansore è duplice: elimina l'informazione invariante per aumentazione (che non dovrebbe finire nelle rappresentazioni utili a compiti a valle) e espande la dimensionalità in modo non lineare, aumentando la capacità della decorrelazione di ridurre dipendenze statistiche di ordine superiore al secondo.Un risultato chiave di Bardes et al. è che la performance in valutazione lineare su ImageNet migliora monotonicamente con d fino a 8192 senza segnali di saturazione.

21.4 Barlow Twins: ridondanza, cross-correlazione e whitening

Barlow Twins (Zbontar, Jing, Misra, LeCun, Deny — ICML 2021) nasce dall'ipotesi di H. Barlow (1961) che il sistema nervoso trasformi segnali sensoriali ridondanti inun codice fattoriale a componenti statisticamente indipendenti. Due rami identici f_θricevono viste Y^A e Y^B dello stesso batch X. La matrice di cross-correlazione normalizzata è definita componente per componente:

$$C_{ij} = \frac{\Sigma_n z^A_{n,i} z^B_{n,j}}{\sqrt{\Sigma_n (z^A_{n,i})^2} \cdot \sqrt{\Sigma_n (z^B_{n,j})^2}} \quad (15.10)$$

La funzione di perdita Barlow Twins è:

$$L_{BT} = \Sigma_i (1 - C_{ii})^2 + \lambda \Sigma_i \Sigma_{j \neq i} C^2_{ij} \quad (15.11)$$

Barlow Twins come approssimazione del whitening. L'obiettivo L_BT = 0 implica C = I_d, la matrice identità. Il whitening standard trasforma z → C^{-1/2}(z − μ)ottenendo una distribuzione con covarianza identità. Barlow Twins approssima questo obiettivo in modo differenziabile: il termine invariante (1-C_{ii})^2 impone che le componenti omologhe delle due viste siano massimamente correlate (e quindi che abbiano la stessa varianza dopo normalizzazione), mentre il termine di ridondanza λΣC^2_{ij} spinge la matrice di cross-correlazione

verso la struttura diagonale, eliminando la ridondanza cross-branch. A differenza del whitening esplicito, che richiede SVD O(d^3), BarlowTwins è O (nd^2) nel calcolo di C.

Differenze chiave con VICReg. (1) VICReg applica la regolarizzazione a ciascun branch separatamente (v e c su Z e su Z' indipendentemente); Barlow Twins agisce congiuntamente tramite C. (2) VICReg non richiede pesi condivisi né architetture identiche; Barlow Twins prevede rami identici con pesi condivisi. (3) VICReg ha un termine di varianza esplicito che stabilizza il training iniziale, assente in Barlow Twins.(4) Barlow Twins usa un solo iperparametro λ, VICReg ne usa tre.

21.5 Regolarizzazione della variabile latente: metodi e strutture geometriche

Discretizzazione. Se $z \in \{0,...,K-1\}$, la varietà $\Psi(s_x)$ contiene al più K punti.La superficie energetica ha K pozzi discreti e le regioni di Voronoi attorno ad essi partizionano lo spazio. Questa struttura è quella dei modelli VQ-VAE.

Bassa dimensionalità. Se $z \in \mathbb{R}^{d_z}$ con $d_z \ll d$ e Pred è differenziabile,$\Psi(sx)$ ha dimensione $\leq d_z$ in $\mathbb{R}^d$. Il rango della Jacobiana J_z Pred(s_x, z) misura la dimensione effettiva della varietà in ogni punto.

Sparsità L1. Una regolarizzazione $R(z) = \alpha\Sigma_i|z_i|$ induce sparsità in ż.Se $k \leq d_z$ componenti sono attive, $\Psi(s_x)$ è approssimata dall'unione di sottospazi k-dimensionali. Per Pred lineare, Pred(s_x, z) = As_x + Bz, la struttura è identica al dictionary learning: $\Psi(s_x) = \{As_x + Bz : \|z\|_1 \leq \delta\}$.

Regolarizzazione variazionale. Se $z \backsim q(z|x,y)$, la regolarizzazione è$R(z) = KL(q(z|x,y) \| p(z))$, dove p(z) = N(0, I) è la prior gaussiana. Questo è il principio della VAE, e l'obiettivo marginale diventa l'ELBO:$F_w(x,y)$ =

−E_q[log p(y|x,z)] + KL(q(z|x,y) ‖ p(z)).Il termine KL limita la quantità di informazione su y che z può trasportare, implementando il Criterio IV.

Capitolo 22 — I-JEPA, V-JEPA e MC-JEPA: varianti architetturali e risultati sperimentali

22.1 I-JEPA: Self-Supervised Learning da immagini con mascheramento semantico

I-JEPA (Assran, Duval, Misra, Bojanowski, Vincent, Rabbat, LeCun, Ballas — CVPR 2023)è la prima istanziazione rigorosa di JEPA su immagini statiche. La proposta dimostra che rappresentazioni di alta qualità semantica sono apprendibili senza augmentazionimanuali, semplicemente imponendo una struttura di predizione nello spazio delle rappresentazioni.L'architettura si compone di tre reti: context encoder f_θ (ViT), target encoder f_{θ̄} (ViTaggiornato per EMA), e predittore g_φ (narrow ViT).

Un'immagine y è divisa in N = H/P × W/P patch non sovrapposte di dimensione P×P.Per 224×224 con P = 14: N = 256. Il target encoder processa l'immagine completa producendo le rappresentazioni patch-level s_y = f_{θ̄}(y) = {s^y_1,...,s^y_N} ∈ ℝ^{N×d}.I pesi θ̄ vengono aggiornati come media mobile esponenziale:

$$\bar{\theta} \leftarrow m \cdot \bar{\theta} \; + \; (1 \; - \; m) \cdot \theta, \;\; m \approx 0.996 \quad (16.1)$$

Strategia di mascheramento multi-blocco. Vengono campionati M = 4 blocchi target B_1,...,B_M con scala s_i ∽ U(0.15, 0.20) e rapporto d'aspetto r_i ∽ U(0.75, 1.5).I target patch sono s_y(i) = {s^y_j}_{j∈B_i}. Il blocco contestuale B_x viene campionato con scala ∽ U(0.85, 1.0); le sue regioni sovrapposte con i target vengono rimosse.Il context encoder processa solo le patch visibili:

$$s_x = f_\theta(\{patch\_j\}_{j \in B\_x \setminus U\_i B\_i}) \quad (16.2)$$

La funzione di perdita. La predizione del blocco target i è s̃_y(i) = g_φ(s_x, {m_j}_{j∈B_i}),dove m_j = v + pe(j) con v vettore learnable condiviso e pe(j) embedding posizionale 2D sinusoidale della posizione j. La perdita è:

$$L = \frac{1}{M} \Sigma_{i=1}^{M} \Sigma_{j \in B_i} \| \tilde{s}^y_j - s^y_j \|_2^2 \quad (16.3)$$

16.1.1 Perché blocchi grandi inducono semantica: dimostrazione per assurdo

La scala dei blocchi target è il fattore critico per la qualità semantica. Per una prova intuitiva per assurdo: supponiamo che i target siano patch singole (scala ≈ 1/N ≈ 0.004).In questo caso il predittore deve stimare la rappresentazione di un patch 14×14px dal contesto. A questa scala, le strutture dominanti sono bordi, gradienti di luminanza,texture locali: feature di basso livello catturate già dai primi strati del ViT. Il segnale di addestramento non fornisce incentivo al context encoder per costruire rappresentazioni semantiche di oggetti o scene. Il predittore impara a stimare le feature locali per interpolazione spaziale, e il context encoder non impara nulla di semanticamente rilevante.

Con blocchi di scala 0.15–0.20 (70–84 pixel per lato), ogni target contiene una parte semanticamente significativa di un oggetto. Predire la rappresentazione di una tale regione richiede che il context encoder abbia compreso il contenuto globale dell'immagine:la forma, la categoria, le relazioni spaziali tra le parti. Questo è verificato dall'esperimento di ablazione di Assran et al.: ridurre la scala massima dei target da 0.20 a 0.05 causa un calo di 7–9 punti in accuratezza di classificazione lineare su ImageNet-1K, mentre la performance su compiti di basso livello (stima della profondità, corrispondenza di keypoint) rimane stabile, confermando che l'encoder apprende gerarchie di feature.

16.1.2 Risultati sperimentali: confronto sistematico

Su ImageNet-1K in valutazione lineare: I-JEPA ViT-H/14 (300 epoche, < 1200 GPU-h A100)raggiunge 81.0% top-1. Confronto: MAE ViT-H/14 (1600 ep, ~11000 GPU-h) 75.8%, data2vecViT-H (800 ep) 78.9%, iBOT ViT-H/14 (1600 ep) 79.5%. Il rapporto efficienza/prestazioni è quindi superiore di circa un fattore 10 rispetto a MAE. Su valutazione semi-supervisionato con l'1% delle etichette: I-JEPA 64.5% vs MAE 60.8% vs iBOT 62.5%.

Su compiti di trasferimento a basso livello il vantaggio è ancora più netto. Sull'object counting (NSVQ-Count benchmark), I-JEPA produce un errore assoluto medio MAE = 4.8vs iBOT 6.2 vs MAE 7.1. Sulla stima della profondità monoculare (NYU Depth v2, kNN evaluation): I-JEPA REL = 0.14 vs iBOT 0.17 vs MAE 0.21. Questi risultati dimostrano che I-JEPA impara una rappresentazione geometricamente strutturata che codifica la profondità e le relazioni spaziali senza supervisione esplicita.

22.2 V-JEPA: predizione latente video e comprensione del movimento

V-JEPA (Meta AI, 2024) estende il principio di I-JEPA al dominio video. La sfida principale è la ridondanza temporale: frame consecutivi sono quasi identici, il che facilita il mascheramento estremo ma rende banale la predizione se il context encoder può sfruttare la continuità temporale. La soluzione di V-JEPA è la mascheratura che copre l'intera dimensione temporale, impedendo l'uso della ridondanza temporale per la predizione.

16.2.1 Architettura e tokenizzazione 3D

Il backbone è un ViT con joint space-time attention (ViT-L o ViT-H). Un clip di 16 frame campionati con stride temporale 4 (coprendo 64 frame

≈ 2 secondi a 30fps)viene ridimensionato a 224×224. La tokenizzazione usa una convoluzione 3D con filtri 2×16×16, stride temporale 2 e spaziale 16:

$$16 \times 224 \times 224 \times 3 \longrightarrow 8 \times 14 \times 14 \times d \longrightarrow 1568\ token \in \mathbb{R}^{1568 \times d} \quad (16.4)$$

A cui vengono aggiunti embedding posizionali sinusoidali 3D (separabili: pe_3D(t,h,w) =pe_T(t) ⊕ pe_H(h) ⊕ pe_W(w)).

16.2.2 Strategia di mascheramento 3D Multi-Block

Per ogni clip vengono campionate due maschere indipendenti. La maschera short-range è l'unione di 8 blocchi spaziali con scala 0.15 e rapporto d'aspetto ∽ U(0.75,1.5),replicata su tutti gli 8 time step. La maschera long-range usa 2 blocchi con scala 0.70,anch'essa replicata temporalmente. La replicazione temporale è cruciale: impedisce al context encoder di sfruttare la ridondanza temporale per inferire il contenuto dei token mascherati. Il rapporto di mascheramento medio è ≈90%, lasciando al context encoder solo il 10% dei token.

16.2.3 Funzione di perdita L1 e multi-mask prediction

Sia x^L la sequenza di L=1568 token del clip. Data una maschera M che seleziona M token(indici i_1,...,i_M), il context encoder processa i token visibili x^N = (x_{j_1},...,x_{j_N}):

$$z\hat{}N = E_\theta(x\hat{}N) \quad (16.5)$$

Il predittore P_φ riceve z^N e mask tokens m^M = {m_{i_k}} = {v + pe_3D(i_k)}:

$$ś\hat{}M = P_\varphi(z\hat{}N,\ m\hat{}M) \quad (16.6)$$

Il target encoder processa il video completo s^L = E_{θ̄}(x^L). La perdita è:

$$Loss = \frac{1}{M}\Sigma_{k\in\{i_1,\ldots,i_M\}} \|\acute{s}_k - s_k\|_1 \quad (16.7)$$

La norma L1 è preferita alla L2 per la sua robustezza agli outlier nelle rappresentazioni video: singoli token con valori estremi influenzano linearmente la perdita invece che quadraticamente. La multi-mask prediction ammortizza il costo computazionale: per un dato clip, il target encoder viene eseguito una sola volta (costo fisso), mentre il context encoder e il predittore vengono eseguiti separatamente per le due maschere (costo aggiuntivo ≈20%, poiché il context encoder processa solo il 10% dei token).

16.2.4 Risultati sperimentali e scalabilità

V-JEPA ViT-g/14 (1B parametri) addestrato su 2M video pubblici. Protocollo frozen evaluation: backbone congelato, singolo strato cross-attention addestrabile per compito.Su Something-Something-v2 (SSv2 — comprensione causale del movimento): 71.2% top-1 vs VideoMAEv2 61.4% (+9.8pp), DINOv2 49.3% (+21.9pp), OpenCLIP 38.9% (+31.3pp). Su Kinetics-400(riconoscimento azione da apparenza): 82.1% top-1 vs VideoMAEv2 77.9% (+4.2pp). Su ImageNet senza fine-tuning su immagini: 77.9%, superiore di 6pp al miglior modello video precedente.La scalabilità con i dati è un risultato chiave: aumentare il dataset da 500K a 2M video migliora SSv2 di 2–3pp e K400 di 1–2pp senza saturazione, suggerendo che modelli ancora più grandi addestrati su scale web potranno migliorare ulteriormente.

22.3 MC-JEPA: apprendimento congiunto di moto e contenuto

MC-JEPA (Bardes, Ponce, LeCun — 2023) unifica l'apprendimento di feature di contenuto(identificazione di oggetti) e feature di moto (flusso ottico denso) all'interno di un unico encoder condiviso. La motivazione è

che i due obiettivi sono complementari: la comprensione del moto richiede di identificare gli stessi oggetti in frame diversi, il che implica comprensione del contenuto; il contenuto di un oggetto in moto è più ricco se se ne comprende la dinamica.

16.3.1 M-JEPA: stima gerarchica del flusso ottico

Il flusso ottico tra due frame I_t, I_{t+1} ∈ ℝ^{3×H×W} è la mappa di corrispondenza f ∈ ℝ^{2×H×W}. L'encoder produce piramidi di feature a L livelli. Al livello più basso si stima il flusso iniziale, poi raffinato iterativamente:

$$f^{(l+1)}_{t,t+1} = F_\theta(X^{(l)}_t, X^{(l)}_{t+1}, f^{(l)}_{t,t+1}) \quad (16.8)$$

L'estimatore F_θ(X_t, X_{t+1}, f) calcola: (1) warp di X_t con f per ottenere X̃_{t+1};(2) volume di correlazione 4D V = X̃^T_{t+1} X_{t+1} ∈ ℝ^{h×w×h×w}; (3) stima delflusso locale tramite una piccola CNN. La perdita fotometrica è L_{photo} = Σ_p ϱ(I_t(p+f(p)) − I_{t+1}(p)),dove ϱ è la funzione di Charbonnier (surrogato robusto della norma L1). La perdita dismoothness è L_{smooth} = Σ_p(|∂f/∂x| + |∂f/∂y|). La consistenza ciclo avanti-indietroè L^(l)_{cycle} = Σ_p ‖X^(l)_t(p) − f_{t,t+1}(f_{t+1,t}(X^(l)_t(p)))‖_1 ad ogni livello l.

16.3.2 Il multi-task e il ruolo stabilizzante di VICReg

La componente di apprendimento del contenuto applica VICReg a ogni livello della piramide di feature. La perdita totale è:

$$L_{total} = L_{photo} + \alpha L_{smooth} + \beta \Sigma_l L^{(l)}_{cycle} + \gamma \Sigma_l L^{(l)}_{VICReg} \quad (16.9)$$

Senza la regolarizzazione VICReg il training multi-task è instabile: il segnale di flusso domina e collassa le feature verso rappresentazioni di basso livello dipendenti dal movimento. VICReg agisce come ancora

semantica: mantiene le feature di ogni livello diversificate e informative. Risultati sperimentali: Sintel Clean EPE=2.33 (vs UFlow 2.50), KITTI F1=11.2%. Su segmentazione semantica (Pascal VOC, Cityscapes, ADE20k), MC-JEEP supera sia i metodi specializzati in flusso che quelli puramente di content learning,confermando la sinergia dei due obiettivi.

Capitolo 23 — JEPA contro i modelli generativi: confronto formale

23.1 Il tradeoff informativo tra predizione e generazione

La differenza fondamentale tra JEPA e i modelli generativi come MAE può essere formalizzata in termini di informazione mutua. Sia x il contesto osservato e y il target.Un modello generativo come MAE massimizza I (s_y; y) imponendo che s_y sia sufficiente a ricostruire y: la perdita di ricostruzione D(y, Dec(s_y)) = 0 implica s_y = y a meno di isometrie, e quindi I (s_y; y) = H(y). Ma H(y) include informazione irrilevante:la texture stocastica, gli artefatti JPEG, il rumore del sensore. JEPA massimizza I(s_y; y)soggetto al vincolo che s_y sia predicibile da s_x:

$$max\, I(s_y;\, y)\ \ s.t.\ \ D(s_y,\, Pred(s_x,\, z\,*)) \leq \varepsilon,\quad I(z;\, y \mid x) \leq \delta \quad (17.1)$$

Questi vincoli selezionano le componenti di y predicibili da x, eliminando quelle stocastiche e irrilevanti. Il risultato è l'informazione mutua predicibile I(s_y; y ∣ x): una quantità più strettamente allineata con la comprensione semantica rispetto alla semplice informazione totale I (s_y; y).

23.2 Costo computazionale della predizione in spazio pixel

La predizione in spazio pixel ha un costo proporzionale a dim(Y). Il gradiente della perdita di ricostruzione D(y, Dec(s_y)) rispetto ai parametri del decoder ha dimensione proporzionale a dim(Y) = H×W×C. In JEPA, il gradiente rispetto al predittore è proporzionale ad $\ll$ dim(Y). Il rapporto di riduzione è:

$$r = \frac{dim(Y)}{d} = \frac{H \times W \times C}{d} = \frac{224 \times 224 \times 3}{1280} \approx 118 \quad (17.2)$$

Questo spiega perché I-JEPA addestri un ViT-H/14 in ~1200 GPU-ore mentre MAE richiede~11000 GPU-ore con lo stesso backbone e prestazioni inferiori: la differenza non è solo nel costo del decoder (che MAE ha e I-JEPA no), ma anche nella dimensione dello spazio di predizione, che in I-JEPA è circa 118 volte più compatto.

23.3 Rappresentazioni di MAE vs I-JEPA: analisi spettrale

Le analisi spettrali di He et al. (2022) mostrano che le rappresentazioni di MAE hanno uno spettro di singolari concentrato sulle prime componenti (ad alta varianza), con una "coda" lunga di componenti a bassa varianza. Questo spettro indica che MAE apprende rappresentazioni in cui poche dimensioni catturano la maggior parte della varianza(le strutture visive ricorrenti), mentre molte dimensioni codificano dettagli specifici dell'istanza (texture, rumore). Le rappresentazioni di I-JEPA, al contrario, mostrano uno spettro più uniforme, con un maggior numero di componenti significative, indicando che I-JEPA distribuisce l'informazione semantica su più dimensioni in modo più efficiente.

Questo si traduce in vantaggio pratico: per compiti come l'object counting, dove l'informazione numerica è distribuita sull'intera immagine (non si concentra in particolari strutture visive), le rappresentazioni distribuite di I-JEPA catturano meglio l'informazione rilevante rispetto alle rappresentazioni concentrate di MAE. Per compiti come la classificazione semantica, dove le strutture globali dominano, entrambe le architetture sono competitive.

23.4 Il collasso nelle architetture generative e in JEPA

Nei modelli generativi come MAE, il collasso della rappresentazione è strutturalmente prevenuto dal decoder: se l'encoder producesse una costante, il decoder non potrebbe ricostruire con fedeltà. Ma questa salvaguardia ha un costo: il decoder può apprendere a ricostruire y usando informazioni di basso livello, catturate già dai primi strati.Le rappresentazioni risultanti sono utili per fine-tuning end-to-end ma meno adatte per la valutazione frozen.

In JEPA, il rischio principale è il collasso di predizione: target encoder e context encoder potrebbero convergere verso rappresentazioni identiche. Questo è prevenuto dall'aggiornamento EMA del target encoder: θ evolve più lentamente di θ, mantenendo una differenza che fornisce segnale di addestramento non triviale. La stabilità teorica di questo schema è dimostrata da Grill et al. (2020) per BYOL e da Richemond et al.(2020) per varianti di BYOL: il target encoder stabilizza il training comportandosi come un "teacher stabile" che il context encoder cerca di emulare.

23.5 Modelli generativi: capacità mancante in JEPA e risposta di LeCun

JEPA rinuncia alla capacità generativa: non può campionare y dallo spazio degli ingressi.Questo è rilevante per applicazioni creative (generazione di immagini e video), data augmentation e per il planning in mondi modello con rappresentazioni concrete. La posizione di LeCun al riguardo è esplicita: un agente intelligente non ha bisogno di immaginare ogni pixel del futuro; ha bisogno di capire cosa accadrà a livello astratto.Un pilota che prevede l'esito di una manovra di sorpasso non visualizza ogni fotogramma della scena futura: costruisce una rappresentazione astratta delle traiettorie, delle distanze,delle velocità. JEPA, modellando il futuro nello spazio delle rappresentazioni, si avvicina a questa forma di cognizione più dei modelli pixel-level.

23.6 JEPA e Diffusion Models: Complementarità e Integrazione

Il confronto tra JEPA e i modelli di diffusione è particolarmente rilevante nel contesto della robotica, dove i modelli di diffusione (Capitolo 14) sono diventati la scelta dominante per la generazione di policy (Diffusion Policy, Chi et al. 2023). Un modello di diffusione per le azioni apprende p(a | o) come processo di de-noising iterativo: a^{(0)} ~ p(a | o) è campionato iterando a^{(k-1)} = f_θ(a^{(k)}, k, o) partendo da rumore gaussiano a^{(K)} ~ N(0,I). Rispetto a JEPA, che modella la struttura latente senza generare esplicitamente, un modello di diffusione per le azioni produce distribuzioni multimodali sulle azioni (utile per task con più soluzioni plausibili) ma richiede K passi di de-noising per ogni azione generata (tipicamente K = 100), introducendo una latenza non trascurabile per il controllo in tempo reale.

L'integrazione più promettente combina un encoder JEPA per la rappresentazione percettiva con un modello di diffusione per la generazione delle azioni: z = Enc(o) è calcolato con un backbone JEPA addestrato con obiettivi non contrastivi (VICReg, Barlow Twins), e p(a | z) è modellata da un modello di diffusione condizionato sul latente z. Questa architettura, realizzata da sistemi come π_0 (Black et al. 2024) e HiRT (Team, 2024), disaccoppia la qualità della rappresentazione (responsabilità dell'encoder JEPA) dalla qualità della policy (responsabilità del diffusion head), permettendo di pre-addestrare l'encoder su dati non etichettati a grande scala e di fine-tunare la policy su dati di dimostrazione relativamente scarsi.

23.7 Limiti Formali di JEPA e Prospettive

Nonostante i vantaggi computazionali e la qualità delle rappresentazioni, JEPA presenta limiti formali ben definiti. Il principale è l'impossibilità di generare campioni dalla distribuzione dell'input: data una

rappresentazione latente z, non esiste un decoder che ricostruisce in modo fedele l'osservazione corrispondente. Questo limita l'applicabilità di JEPA in task che richiedono generazione visiva (video prediction, image synthesis, data augmentation). LeCun riconosce questo limite e lo considera un design choice deliberato: JEPA non è un modello generativo e non aspira a esserlo. La conseguenza pratica è che sistemi basati esclusivamente su JEPA non possono essere usati per la generazione di dati sintetici di addestramento, che è invece una delle applicazioni più promettenti dei modelli di diffusione in robotica.

Un secondo limite di JEPA è la dipendenza dalla qualità della maschera di predizione. In I-JEPA, le regioni target sono patch casuali dell'immagine; in V-JEPA, i tube spaziotemporali. La scelta della strategia di masking influenza fortemente la struttura delle rappresentazioni apprese: maschere troppo piccole portano a un predittore banale (interpolazione locale); maschere troppo grandi portano a un collasso su rappresentazioni globali non localizzabili. L'ottimizzazione automatica della strategia di masking è un problema aperto: i lavori recenti (MAR, Li et al. 2024; FlexMask, 2025) esplorano masking adattativo che si aggiusta in base alla difficoltà del task, avvicinando JEPA al paradigma del curriculum learning. Per la robotica incarnata, la direzione più promettente è il masking basato sull'azione: mascherare le regioni dell'immagine che cambiano a seguito di un'azione, forzando il predittore a modellare le conseguenze delle azioni — esattamente la struttura del world model attivo discusso nel Capitolo 16.

Capitolo 24 — JEPA e la robotica embodied: modelli del mondo per agenti fisici

24.1 Perché la robotica ha bisogno di modelli del mondo appresi

La robotica moderna affronta una sfida epistemica che non ha equivalente in altri campi dell'intelligenza artificiale: un agente fisico deve agire in un mondo continuo,tridimensionale, parzialmente osservabile e fisicamente vincolato, con un corpo che deve essere controllato in tempo reale. Le soluzioni classiche — pianificatori geometrici basati su modelli analitici, controller PID, planning con SLAM — funzionano in ambienti strutturati e noti a priori, ma falliscono sistematicamente in presenza di oggetti sconosciuti, superfici deformabili, occlusion dinamica o cambiamenti imprevisti nella struttura dell'ambiente.

La risposta dell'apprendimento automatico alla robotica — il reinforcement learning profondo — ha mostrato risultati impressionanti in giochi digitali (Atari, Go, StarCraft)ma fatica nella robotica fisica per due ragioni fondamentali. Prima: il RL model-free richiede milioni di interazioni con l'ambiente reale, che sono lente, costose e potenzialmente distruttive per il robot. Seconda: il segnale di reward scalare è informazione di bassa qualità: un robot che cerca di afferrare un oggetto riceve un segnale binario (successo/fallimento)che non specifica quale aspetto dell'esecuzione deve migliorare.

Il model-based RL tenta di aggirare la prima limitazione costruendo un modello del mondo che consenta al robot di pianificare senza interagire fisicamente con l'ambiente(world model planning). Ma i

modelli del mondo classici hanno due debolezze: (a) degenerativi, devono predire ogni pixel del futuro, diventando computazionalmente costosi e fragili in ambienti caotici; (b) se appresi con supervisione, richiedono enormi quantità di dati etichettati. JEPA offre una terza via: un modello del mondo non generativo, appreso in modo auto supervisionato, che predice il futuro nello spazio astratto delle rappresentazioni invece che nello spazio dei pixel.

24.2 Il world model robotico come JEPA: formalismo

Sia s ∈ S lo spazio degli stati del robot e del suo ambiente (stato propriocettivo +percezione visiva), a ∈ A lo spazio delle azioni (comandi motori, forze, velocità).Un world model robotico è una funzione Pred : S × A × Z → S che predice lo stato futuro dato lo stato corrente, l'azione e una variabile latente z che rappresenta l'incertezza sull'evoluzione del mondo. In forma JEPA:

$$\tilde{s}[t+1] = Pred(s[t],\ a[t],\ z[t]), \quad z[t] \backsim p(z \mid s[0:t]) \quad (18.1)$$

La chiave è che s è una

24.3 Pianificazione differenziabile con world model JEPA

Il Model-Predictive Control (MPC) con orizzonte T è la forma di pianificazione adottata nell'architettura di LeCun (Mode-2). Dato lo stato corrente s[0], si trovano le azioni che minimizzano il costo totale previsto:

$$(\check{a}[0:T]) = argmin_{a[0:T]} \Sigma_{t=0}^{T-1} C(s[t+1]), \quad s[t+1] = Pred(s[t],\ a[t],\ z[t]) \quad (18.2)$$

Poiché Pred e C sono funzioni differenziabili, il gradiente ∂L/∂a[t] può essere calcolato per backpropagation attraverso la sequenza di

predizioni, come nel differentiable model predictive control (differentiable MPC, DMPC). Il gradiente è:

$$\frac{\partial L}{\partial a[t]} = \Sigma_{\tau=t}^{T-1} \frac{\partial C(s[\tau+1])}{\partial s[\tau+1]} \cdot \frac{\partial s[\tau+1]}{\partial a[t]} \quad (18.3)$$

Il secondo termine ∂s[τ+1]/∂a[t] richiede la propagazione del gradiente attraversoτ-t passi del world model: questo è l'equivalente della backpropagation through time (BPTT)nel planning. Per τ-t grande, i gradienti possono avere il problema del vanishing/exploding(come nelle RNN). Le soluzioni pratiche includono: clip del gradiente, truncated BPTT (propagazione solo sugli ultimi κ passi), e l'uso di world model con struttura residuale(s[t+1] = s[t] + f(s[t], a[t])) che facilita la propagazione del gradiente.

24.4 Apprendimento della rappresentazione robot con JEPA

Per un robot che apprende a manipolare oggetti, la rappresentazione dell'osservazione è fondamentale: deve catturare informazioni rilevanti per il compito (posizione dell'oggetto,orientazione, forma, proprietà della superficie) e ignorare informazioni irrilevanti(illuminazione, texture del piano di lavoro, variazioni di colore). JEPA, addestrato su sequenze di osservazioni del robot in interazione con oggetti, impara automaticamente a costruire tali rappresentazioni.

Setup di apprendimento. Il robot raccoglie dati di esplorazione autonoma in modalità passiva: muove il braccio in traiettorie casuali, filmando l'ambiente con camere RGB-D e registrando i dati propriocettivi. I dati vengono usati per addestrare un JEPA video con M-JEPA per il flusso ottico (che cattura le relazioni cinematiche tra i movimenti del braccio e gli spostamenti degli oggetti) e VICReg per le feature semantiche (che catturano le proprietà degli oggetti indipendenti dal punto di vista).

Risultati nella letteratura. Pari et al. (2021) e Srinivas et al. (2018) hanno mostrato che metodi di self-supervised learning applicati a rappresentazioni visual-motorie migliorano significativamente l'efficienza del model-based RL su compiti di manipolazione(raggiungimento, presa, push), riducendo il numero di interazioni necessarie di un fattore 5–10 rispetto al RL model-free. Yarats et al. (2021) dimostrano che encoder pre-addestrati con SSL su dati robot migliorano la qualità del world model, riducendo il planning horizon necessario per risolvere compiti di manipolazione complessi.

24.5 Il modello ego-robot come caso speciale di JEPA

LeCun propone di separare il world model in due componenti: un

ego-model (modello del corpo del robot) e un world-model (modello dell'ambiente esterno). L'ego-model predice l'evoluzione dello stato propriocettivo del robot (angoli articolari, velocità, posizione dell'end-effector) a partire dai comandi motori. Il world-model predice l'evoluzione dell'ambiente esterno dato lo stato dell'ego-model e la sua interazione con gli oggetti. La separazione è motivata dalla diversa predicibilità dei due componenti: l'ego-model è quasi deterministico(dati i comandi, la cinematica del robot è nota), mentre il world-model è stocastico(la risposta degli oggetti all'interazione può essere incerta).

Formulazione JEPA dell'ego-model. Siano q[t] ∈ ℝ^{n_{dof}} gli angoli articolari,̇q[t] le velocità, τ[t] i comandi di coppia. L'ego-model predice:

$$q[t+1] = q[t] + \Delta t \dot{q}[t] + F_\theta(q[t], \dot{q}[t], \tau[t]) \quad (18.4)$$

dove F_θ è una rete neurale che apprende le dinamiche residue rispetto alla cinematica di secondo ordine q[t+1] = q[t] + Δt ̇q[t] (modello lineare di default). Addestrato sudati propriocettivi del robot in

movimento libero (senza contatto con oggetti), F_θapprende le non-linearità cinematiche, la flessibilità delle giunzioni e la frizione.Il modello risultante è molto più accurato del modello geometrico rigido, con errori di predizione tipicamente nell'ordine del millimetro per orizzonti di 0.5 secondi.

World model per la manipolazione. Il world model predice la posizione futura degli oggetti dato lo stato dell'ego-model e il contatto. In forma JEPA, data una rappresentazione visiva dell'oggetto s_{obj}[t] (posizione 3D, orientazione, forma)e dello stato di contatto c[t] (forza, punto di contatto, area di contatto):

$$s_{obj}[t+1] = Pred_w(s_{obj}[t],\ c[t],\ z[t]) \quad (18.5)$$

La variabile latente z[t] rappresenta l'incertezza sulla risposta dell'oggetto al contatto:se l'oggetto è rigido z è quasi deterministico; se è deformabile (ad esempio un panno),z cattura le diverse modalità di deformazione possibili. Addestrare Pred_w su dati di manipolazione robotica è computazionalmente costoso in modo model-free, ma può essere reso efficiente con JEPA pre-addestrato su video di manipolazione umana, sfruttando il transfer da dati visivi egocentrici di YouTube, EPIC-Kitchens, o HOI4D.

PARTE VI

Sintesi e Frontiere

Sull'unità concettuale dei modelli predittivi nel latente, le sfide aperte della causalità e della composizionalità, e le prospettive per una robotica cognitiva fondata su rappresentazioni astratte dell'azione.

Capitolo 25 — Integrazione: World Models, JEPA e Robotica

25.1 Convergenza Epistemica: il Predittivo nel Latente come Paradigma Unificante

Le tre architetture al centro di questa monografia — i world model nella formulazione RSSM-Dreamer, l'architettura JEPA di LeCun, e i sistemi di controllo robotico basati sull'apprendimento — sono state sviluppate da comunità di ricerca in parte distinte, con motivazioni parzialmente diverse e con un lessico tecnico che riflette tali differenze. Eppure, una lettura attenta del loro formalismo matematico rivela un'unità di struttura che non è superficiale: tutte e tre convergono verso il paradigma della predizione nel latente come meccanismo fondamentale di apprendimento e pianificazione.

La tesi che questo capitolo intende argomentare in modo formale è la seguente: JEPA e i world model del tipo RSSM sono due istanze di una medesima classe astratta di architetture predittive nel latente, che differiscono essenzialmente per due aspetti — la presenza o assenza di un termine di ricostruzione nell'obiettivo di addestramento, e la presenza o assenza di un modello esplicito delle azioni come variabile condizionante. Comprendere questa relazione non è meramente un esercizio di tassonomia: apre la strada a una genuina integrazione delle due famiglie di architetture in sistemi robotici che combinano la ricchezza rappresentazionale dei world model con l'efficienza computazionale e la robustezza delle predizioni nel latente di JEPA.

25.2 Formalizzazione della Classe Unificante

Il framework delle architetture predittive nel latente

Sia X lo spazio delle osservazioni (immagini RGB, flussi sensoriali, misure propriocettive) e sia $Z = \mathbb{R}^{d}z$ lo spazio latente di dimensione d_z. Una architettura predittiva nel latente (Latent Predictive Architecture, LPA) è caratterizzata dai seguenti componenti:

Un encoder contesto $\varepsilon_x : X \to Z$ che mappa l'osservazione contesto x in una rappresentazione latente $s_x = \varepsilon_x(x)$. Un encoder target $\varepsilon_y : X \to Z$ — distinto o identico ad ε_x a seconda della variante architetturale — che mappa l'osservazione target y in $s_y = \varepsilon_y(y)$. Un predittore $Pred : Z \times A \to Z$ che, data la rappresentazione del contesto e un'azione a (eventualmente nulla), produce la predizione della rappresentazione target $\tilde{s_y} = Pred(s_x, a; z)$ dove $z \in Z_{lat}$ è la variabile latente che copre l'incertezza sullo stato non osservato. La funzione obiettivo di addestramento minimizza la discrepanza tra la rappresentazione target predetta e quella prodotta dall'encoder target:

$$\mathcal{L}\_LPA = \mathbb{E}\_{(x,y,a)\sim\mathcal{D}} [\, D(Pred(\varepsilon_x(x), a ; z),\ \varepsilon\_y(y)) \,] + \lambda\, R(\varepsilon_x, \varepsilon\_y, Pred) \quad (19.1)$$

dove $D : Z \times Z \to \mathbb{R}^{\geq 0}$ è una misura di discrepanza nello spazio latente, $\lambda > 0$ è un coefficiente di regolarizzazione e R è un termine di regolarizzazione che previene il collasso rappresentazionale. La funzione D può essere la distanza euclidea al quadrato $\|\cdot\|^2$ (come in JEPA), la divergenza di Kullback–Leibler (come nei VAE), o qualsiasi altra funzione di divergenza coerente.

RSSM come caso speciale di LPA con ricostruzione

Il Recurrent State Space Model della famiglia Dreamer è un caso speciale di LPA in cui il termine di regolarizzazione R include un obiettivo di ricostruzione dell'osservazione e una divergenza KL tra prior e posterior. Scritto nell'unificazione LPA, l'obiettivo di addestramento dell'RSSM è:

$$\mathcal{L}_{RSSM} = -\mathbb{E}_q[\, \Sigma_t \log p_\theta(o_t \mid s_t) \,] + \beta\, KL[\, q_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t) \,\|\, p_\theta(s_t \mid s_{t-1}, a_{t-1}) \,] \quad (19.2)$$

Il primo termine è il termine di ricostruzione: esso richiede che il decoder $p_\theta(o_t \mid s_t)$ produca l'osservazione o_t a partire dallo stato latente s_t, e serve come segnale di supervisione per costruire rappresentazioni latenti informative. Il secondo termine forza il posterior ammortizzato $q_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$ a coincidere con il prior dinamico $p_\theta(s_t \mid s_{t-1}, a_{t-1})$, che è il predittore delle dinamiche latenti. Il prior svolge quindi il ruolo del componente Pred nella formulazione LPA, e la coppia (prior, posterior) implementa la stessa logica della coppia (encoder target, predittore) di JEPA.

JEPA come LPA senza ricostruzione

L'architettura JEPA di LeCun è un caso speciale di LPA in cui il termine di ricostruzione è assente e la discrepanza D è la distanza euclidea al quadrato. In dettaglio, fissate le maschere di contesto e target su una stessa immagine o sequenza, l'obiettivo di I-JEPA (l'istanza per immagini) è:

$$\mathcal{L}_{JEPA} = \Sigma_{(x,y)\in\mathcal{M}} \| \tilde{s_y} - stop\_grad(s_y) \|^2 \quad (19.3)$$

dove $\tilde{s_y} = \text{Pred}(\varepsilon_x(x))$ è la predizione della rappresentazione target a partire dal contesto mascherato, $s_y = \varepsilon_y(y)$ è la rappresentazione target prodotta dall'encoder target (aggiornato con il momentum EMA invece che con il gradiente), e stop_grad indica che il gradiente non fluisce attraverso l'encoder target. L'assenza del termine di ricostruzione è la scelta cruciale che distingue JEPA da tutti i modelli generativi precedenti: il modello non apprende a riprodurre i pixel, ma a predire le caratteristiche semantiche estratte da un encoder. Questo porta a rappresentazioni qualitativamente diverse — più compatte, più semantiche, meno sensibili al rumore a basso livello — rispetto a quelle apprese con un obiettivo di ricostruzione.

25.3 Action-Conditioned JEPA: il Modello del Mondo senza Decodificatore

La convergenza tra JEPA e world model suggerisce naturalmente la costruzione di un'architettura ibrida che combini i punti di forza di entrambe le famiglie: le rappresentazioni semantiche di JEPA con la struttura causale e condizionata dall'azione dei world model. Si definisce pertanto l'Action-Conditioned JEPA (AC-JEPA) come la variante di JEPA in cui il predittore Pred è condizionato sull'azione a

$$\mathcal{L}_{AC\text{-}JEPA} = \mathbb{E}_{(x,a,y)\sim\mathcal{D}} \| Pred(\varepsilon_x(x), \varphi_a(a)) - stop\_grad(\varepsilon_y(y)) \|^2 \quad (19.4)$$

dove $\varphi_a : A \to \mathbb{R}^{d_a}$ è un encoder dell'azione (lineare o non lineare a seconda della struttura dello spazio A) e il predittore Pred è una rete che fonde la rappresentazione del contesto con quella dell'azione. Nella formulazione dinamica temporale, $x = o_t$, $y = o_{t+k}$ per un orizzonte predittivo $k \geq 1$, e il predittore apprende la transizione nelle dinamiche latenti indotta dall'azione a

Il confronto formale tra AC-JEPA e RSSM è illuminante. Entrambi apprendono una funzione di transizione latente condizionata dall'azione; la differenza principale risiede nel regime di addestramento e nell'obiettivo. L'RSSM minimizza la variational lower bound sull'ELBO, che implica sia un termine predittivo (la corrispondenza tra prior e posterior) sia un termine di ricostruzione. AC-JEPA minimizza solo il termine predittivo nel latente, rinunciando alla ricostruzione. Questa rinuncia ha conseguenze profonde: il modello non può essere usato per generare osservazioni sintetiche (e quindi non può essere usato per allucinare rollout osservativi), ma le sue rappresentazioni sono più robuste, più compatte e — come suggeriscono i risultati empirici — più trasferibili a nuovi compiti attraverso l'adattamento della testa di predizione.

25.4 Pianificazione nel Latente senza Decodificatore

Una delle conseguenze più importanti dell'architettura AC-JEPA per la robotica è la possibilità di pianificare interamente nello spazio latente, senza mai ricorrere alla generazione di osservazioni. Nel framework Dreamer, la pianificazione avviene già nel latente: la policy e la funzione valore sono apprese su rollout immaginati nello spazio s, non nello spazio o. Tuttavia, Dreamer richiede che il modello sia stato addestrato con un obiettivo di ricostruzione, il che impone il costo computazionale e la complessità architetturale del decodificatore durante l'addestramento.

Con AC-JEPA, la pianificazione nel latente è accessibile senza aver mai addestrato un decodificatore. Dato uno stato iniziale $s_0 = \varepsilon_x(o_0)$, una sequenza di azioni $\{a_0, a_1, \ldots, a_{H-1}\}$ e il predittore dinamico Pred, si simula la traiettoria latente:

$$s_{t+1} = Pred(s_t, a_t), \quad t = 0, 1, \ldots, H-1 \quad (19.5)$$

La funzione di costo da minimizzare è definita direttamente nel latente: anziché una funzione di ricompensa sulle osservazioni, si utilizza una funzione $r(s_\theta(r_t \mid s_t)$. Il problema di pianificazione diventa

$$a^* = argmax_{a_0,\ldots,a_{H-1}} \Sigma_{t=0}^{H-1} \gamma^t r(Pred^t(s_0, a)) \quad (19.6)$$

dove Pred^t denota l'applicazione iterata di Pred per t passi. Questo problema può essere risolto con Cross-Entropy Method (CEM), con discesa del gradiente attraverso il predittore differenziabile, o con metodi di pianificazione basata su campioni come MPPI (Model Predictive Path Integral). La chiave della praticabilità è la differenziabilità di Pred rispetto alle azioni, che consente di calcolare il gradiente $\partial r/\partial a$ propagando i gradienti attraverso la sequenza di predizioni —

esattamente come Dreamer propaga i gradienti attraverso i rollout immaginati dell'RSSM.

25.5 Gerarchia Temporale e Astrazione dell'Azione

Un aspetto in cui JEPA e world model si complementano è la gestione della gerarchia temporale. L'architettura Hierarchical JEPA (H-JEPA) proposta da LeCun nel documento programmatico «A Path Towards Autonomous Machine Intelligence» ipotizza una struttura a livelli in cui moduli JEPA operano a scale temporali diverse: il livello inferiore predice rappresentazioni latenti a breve orizzonte (singoli frame o segmenti di millisecondi), mentre i livelli superiori predivono rappresentazioni sempre più astratte e a orizzonte sempre più lungo. Questa struttura corrisponde esattamente all'astrazione temporale che si cerca di realizzare nei world model gerarchici e nei framework di apprendimento per rinforzo con opzioni.

La formalizzazione matematica della gerarchia JEPA è la seguente. Si definiscono L livelli di rappresentazione, con encoder ε_1, …, ε_L e predittori Pred_1, …, Pred_L. Il livello l opera su sequenze di rappresentazioni del livello l−1 con un orizzonte temporale $\Delta_l = \Delta_0^l$ (crescita esponenziale dell'orizzonte con il livello). L'obiettivo di addestramento è la somma degli obiettivi per ciascun livello:

$$\mathcal{L}_{\text{H-JEPA}} = \sum_{l=1}^{L} \lambda_l \cdot \mathcal{L}_{\text{JEPA}}^{(l)} \quad (19.7)$$

dove i pesi λ_l bilanciano l'importanza relativa dei diversi livelli. Le azioni per il livello l non sono le azioni primitive dello spazio A, bensì le «opzioni» del livello l−1 — sequenze di azioni primitive che realizzano comportamenti elementari a un livello di astrazione superiore. Questa struttura è formalmente equivalente al framework Options di Sutton, Precup e Singh nel contesto del reinforcement learning gerarchico,

suggerendo una strada naturale per integrare JEPA con le architetture di pianificazione gerarchica.

Capitolo 26 — Sfide Aperte: Causalità, Composizionalità e Divario Simulazione-Realtà

26.1 Il Problema della Causalità nelle Rappresentazioni Apprese

Correlazione e intervento: la distinzione fondamentale

Le architetture descritte nelle parti precedenti di questa monografia — RSSM, JEPA, AC-JEPA — apprendono la propria rappresentazione del mondo minimizzando obiettivi predittivi su dati di esperienza. In tutti i casi, i dati provengono da una distribuzione osservazionale: l'agente osserva traiettorie (o_t, a_t, o_{t+1}) generate da una policy di comportamento (o di raccolta dati) fissa. Questo regime osservazionale è sufficiente per apprendere correlazioni statistiche tra stati, azioni ed esiti, ma non è sufficiente per apprendere la struttura causale del mondo — ovvero la struttura che risponde a domande del tipo: «cosa accadrebbe se eseguissi l'azione a, indipendentemente dalla policy che di solito produce a in questo contesto?»

Il linguaggio formale per rispondere a domande di questo tipo è il calcolo do di Pearl, introdotto nei tardi anni Novanta e sistematizzato nel libro «Causality» del 2000. Il calcolo do introduce l'operatore di intervento do(A = a) che descrive il valore della variabile A quando essa viene fissata esternamente al valore a, rompendo le dipendenze causali provenienti dai genitori di A nel grafo causale. La distribuzione probabilistica indotta dall'intervento è in generale diversa dalla distribuzione osservazionale condizionata:

$$P(Y \mid do(A = a)) \neq P(Y \mid A = a) \quad (20.1)$$

La differenza tra i due termini è precisamente la confusione causale: la distribuzione condizionata $P(Y|A=a)$ include il contributo delle cause comuni di A e Y (i cosiddetti confounders), mentre la distribuzione interventistica $P(Y|do(A=a))$ le isola, fissando A esternamente e rompendo le frecce causali entranti. Per un agente robotico che apprende da dati osservazionali raccolti con una policy fissa, l'errore di confusione si traduce in una sottostima sistematica dell'effetto causale delle proprie azioni.

World Model causali e il do-calculus

Un world model causale è un modello che, oltre alla distribuzione di transizione $P(s_{t+1}|s_t, a_t)$ apprende dall'esperienza, è in grado di rispondere correttamente a query interventistiche del tipo $P(s_{t+k}|do(a_t = a), \ldots, do(a_{t+k-1} = a'))$. Formalmente, un Structural Causal Model (SCM) è una quadrupla (U, V, F, P_U), dove $U = \{U_1, \ldots, U_n\}$ sono le variabili esogene (rumore), $V = \{V_1, \ldots, V_m\}$ sono le variabili endogene (osservabili), $F = \{f_1, \ldots, f_m\}$ sono le funzioni strutturali $V_i = f_i(Pa(V_i), U_i)$ — dove $Pa(V_i)$ è l'insieme dei genitori causali di V_i nel grafo — e P_U è la distribuzione delle variabili esogene.

L'intervento $do(V_i = v)$ è modellato come la modifica del modello strutturale ottenuta sostituendo la funzione f_i con la funzione costante $V_i = v$, producendo il grafo mutilato in cui tutte le frecce causali entranti in V_i sono rimosse. La distribuzione interventistica $P(V \setminus V_i | do(V_i = v))$ si calcola nel modello mutilato. Nell'SCM, la regola di aggiustamento per backdoor consente di identificare la distribuzione interventistica da dati osservazionali quando esiste un insieme di variabili W che soddisfa il backdoor criterion:

$$P(Y \mid do(X=x)) = \Sigma\_w \, P(Y \mid X=x, W=w) \, P(W=w) \quad (20.2)$$

La formula (20.2) è il fondamento del deconfounding osservazionale: se si conosce l'insieme W di confounders (o un

sottoinsieme sufficiente), è possibile stimare l'effetto causale senza mai eseguire esperimenti controllati. La sfida aperta nell'apprendimento automatico è che, in ambienti visivi complessi, le variabili W non sono note a priori e devono essere scoperte dalla struttura dei dati — un problema di causal discovery che rimane in larga misura irrisolto per rappresentazioni ad alta dimensionalità come le immagini.

Rappresentazione causale latente

Un approccio promettente al problema della causalità nelle rappresentazioni latenti è la Causal Representation Learning (CRL), che cerca di apprendere rappresentazioni latenti che corrispondano alle variabili causali del meccanismo generativo sottostante. Formalmente, si assume che i dati $x \in X$ siano generati da un insieme di variabili latenti causali $c = (c_1, \ldots, c_k)$ secondo un meccanismo generativo $x = g(c)$, e che le variabili c obbediscano a un SCM. L'obiettivo della CRL è apprendere un encoder $\varepsilon : X \to Z$ tale che $\varepsilon(x) \approx c$ — ovvero che le coordinate dello spazio latente corrispondano alle variabili causali del meccanismo generativo.

Kügelgen e colleghi (2021) e Schölkopf e colleghi (2021) hanno dimostrato condizioni di identificabilità della CRL: sotto certe ipotesi sulla struttura del grafo causale e sulla distribuzione delle perturbazioni, è possibile identificare le variabili causali latenti a meno di trasformazioni element-wise, ovvero separare le variabili causali ma non necessariamente ordinarle o scalarle. Questi risultati di identificabilità, pur non essendo ancora sufficientemente generali per applicazioni pratiche non supervisionate, forniscono garanzie teoriche che rendono la CRL una direzione di ricerca promettente per la costruzione di world model intrinsecamente causali.

26.2 Composizionalità e il Problema del Binding

Composizionalità strutturale nelle rappresentazioni

La composizionalità è la proprietà per cui il significato di una struttura complessa è determinato dal significato delle sue parti e dalle regole della loro combinazione. Nei linguaggi naturali, la frase «il robot rosso porta la scatola blu» ha un significato che si ottiene componendo le rappresentazioni di «robot», «rosso», «porta», «scatola» e «blu» secondo la struttura sintattica della frase. Il sistema visivo umano possiede una capacità analoga: può riconoscere oggetti mai visti come combinazioni di proprietà note (colore, forma, materiale, posizione relativa) e può ragionare su scene composizionali che non ha mai osservato prima.

Le architetture neurali standard — reti convoluzionali, Transformer — rappresentano la scena visiva in modo distribuito: ogni feature della rappresentazione latente risponde a una combinazione non interpretabile di aspetti della scena, senza che esista una corrispondenza biunivoca tra posizioni del tensore latente e oggetti della scena. Questa modalità di rappresentazione distribuita è eccellente per il riconoscimento ma problematica per il ragionamento composizionale: per rispondere alla domanda «quanti oggetti rossi ci sono nella scena?» con un sistema a rappresentazione distribuita, la risposta deve essere emergente da pattern della rappresentazione globale piuttosto che essere computata direttamente da slot dedicati agli oggetti.

Slot Attention e rappresentazioni oggetto-centriche

Il meccanismo Slot Attention, introdotto da Locatello e colleghi (2020), propone un approccio oggetto-centrico alla rappresentazione: invece di codificare la scena in un singolo vettore latente, la scena viene scomposta in K slot indipendenti, ciascuno dei quali compete per «catturare» le informazioni di uno specifico oggetto nella scena. Il meccanismo di attenzione iterato che distribuisce le informazioni della scena agli slot è definito come segue.

Sia $F \in \mathbb{R}^{N \times d_{feat}}$ il tensore delle feature estratte dall'encoder (N posizioni, d_{feat} canali) e $S \in \mathbb{R}^{K \times d_{slot}}$ la matrice degli slot. Ad ogni iterazione T dell'algoritmo di attenzione, gli slot vengono aggiornati come:

$$A_{kn} = softmax_k(q_k \cdot k_n / \sqrt{d}) \in \mathbb{R}^{K \times N} \quad (20.3)$$

$$A'_{kn} = A_{kn} / \Sigma_{n'} A_{kn'} \qquad (normalizzazione\ per\ slot) \quad (20.4)$$

$$updates_k = \Sigma_n A'_{kn} \cdot v_n \quad (20.5)$$

$$S_k \leftarrow GRU(S_k, updates_k) \quad (20.6)$$

dove $q_k = W^q S_k$, $k_n = W^k F_n$ e $v_n = W^v F_n$ sono le proiezioni delle feature e degli slot negli spazi di query, chiavi e valori. La normalizzazione (20.4), applicata lungo la dimensione degli slot anziché quella delle posizioni (come nel softmax standard), garantisce che gli slot competano per le feature piuttosto che le feature distribuirsi uniformemente tra gli slot. Questo meccanismo competitivo è la chiave della segmentazione emergente: slot diversi tendono a specializzarsi su regioni diverse della scena, producendo naturalmente una scomposizione oggetto-centrica senza supervisione.

Composizionalità e world model oggetto-centrici

L'integrazione della rappresentazione oggetto-centrica con i world model apre la strada ai cosiddetti Object-Centric World Model (OCWM), in cui le dinamiche sono modellate non al livello della scena globale ma al livello degli oggetti individuali. Nel framework OCWM, il modello di transizione latente è:

$$s^{(k)}_{t+1} = f_\theta(s^{(k)}_t, a_t, \{s^{(j)}_t\}_{j \neq k}) \quad k = 1, \ldots, K \quad (20.7)$$

dove $s^{(k)}_t$ è lo stato latente dell'oggetto k al tempo t e il modello di transizione può includere interazioni tra oggetti diversi (il termine $\{s^{(j)}_t\}$).

Questa struttura è compositiva nel senso che il numero di parametri del modello cresce linearmente con il numero massimo di oggetti nella scena, anziché esponenzialmente come richiederebbe la modellazione dell'intera scena senza struttura oggetto-centrica. Quando un nuovo oggetto appare nella scena, un nuovo slot viene allocato e il suo slot latente viene inizializzato, senza dover riaddestrare l'intero modello.

Il limite principale degli OCWM attuali è la difficoltà di ottenere una segmentazione oggetto-centrica robusta in scene visive complesse, con illuminazione variabile, occlusioni parziali e texture simili tra oggetti diversi. I metodi attuali come SLATE e DINOSAUR (2023) hanno migliorato significativamente la qualità della segmentazione integrando feature semantiche estratte da encoder pre-addestrati (come DINO o CLIP) con il meccanismo Slot Attention, ma la robustezza nelle scene del mondo reale rimane un problema aperto.

26.3 Il Divario Simulazione-Realtà

20.3.1 Formalizzazione del divario: POMDP, gap di dinamica e gap di prestazione

Il divario tra simulazione e realtà — noto nella letteratura anglosassone come sim-to-real gap o reality gap — è il fenomeno per cui un sistema di controllo appreso in un ambiente simulato degrada le proprie prestazioni quando trasferito all'ambiente fisico reale. Lungi dall'essere un fenomeno monolitico, il divario origina da molteplici fonti di discrepanza che agiscono a scale e su aspetti del sistema diversi. Una trattazione rigorosa richiede innanzitutto un framework formale che distingua le diverse componenti del divario e le relazioni tra di esse; il framework più adeguato è quello dei Processi Decisionali di Markov Parzialmente Osservabili (POMDP).

Un POMDP è la tupla $\mathcal{M}$ = (S, A, T, R, Z, O, γ), dove S $\subseteq \mathbb{R}$^n è lo spazio degli stati del sistema (configurazioni del robot e dell'ambiente), A $\subseteq \mathbb{R}$^m è lo spazio delle azioni, T : S × A → Δ(S) è il modello di transizione stocastica, R : S × A → $\mathbb{R}$ è la funzione di ricompensa, Z è lo spazio delle osservazioni, O : S → Δ(Z) è il modello di osservazione (sensore), e $\gamma \in$ [0,1) è il fattore di sconto. Il POMDP simulato $\mathcal{M}$_s e quello reale $\mathcal{M}$_r condividono gli spazi S, A, Z, ma differiscono nei modelli di transizione T_s ≠ T_r e di osservazione O_s ≠ O_r. La realtà del gap risiede esattamente in queste divergenze.

Il gap di dinamica e il gap di percezione sono formalizzati come divergenze tra le rispettive distribuzioni. Siano D(·‖·) una misura di divergenza (ad esempio la divergenza di Kullback-Leibler o la distanza di Wasserstein). Il gap di dinamica è la divergenza media tra le distribuzioni di transizione simulate e reali, integrata sul dominio stato-azione del POMDP reale:

$$G_{dyn}(\mathcal{M}_s, \mathcal{M}_r) = \mathbb{E}_{(s,a)_{\mathcal{M}_r}}\left[D\left(T_s(\cdot|s,a) \,\|\, T_r(\cdot|s,a)\right)\right] \quad (20.10)$$

Analogamente, il gap di percezione misura la divergenza media tra i modelli di osservazione simulato e reale, integrata sulla distribuzione degli stati reali:

$$G_{perc}(\mathcal{M}_s, \mathcal{M}_r) = \mathbb{E}_{s_{\mathcal{M}_r}}\left[D\left(O_s(\cdot|s) \,\|\, O_r(\cdot|s)\right)\right] \quad (20.11)$$

Più importante per la pratica è il gap di prestazione, che misura la differenza nel rendimento atteso di una policy π quando eseguita rispettivamente nel simulatore e nella realtà. Dati il ritorno atteso scontato nel simulatore J_s(π) e nella realtà J_r(π), il gap di prestazione è:

$$G_{perf}(\mathcal{M}_s, \mathcal{M}_r, \pi) = \left|J_{\mathcal{M}_s}(\pi) - J_{\mathcal{M}_r}(\pi)\right| \quad (20.12)$$

dove il ritorno atteso in un ambiente $\mathcal{M}$ è definito dalla traiettoria chiusa indotta da π:

$$J_{\mathcal{M}}(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi,\mathcal{M})}\left[\sum_{t=0}^{T} \gamma^t R(s_t, \pi(b_t))\right] \quad (20.13)$$

La distinzione concettuale fondamentale — dimostrata formalmente da Anand et al. (2025) e discussa in dettaglio nella sezione 20.3.5 — è che minimizzare G_dyn (ovvero costruire un simulatore fisicamente fedele) non è né necessario né sufficiente per minimizzare G_perf. Per sistemi stocastici, è possibile costruire un simulatore con G_dyn arbitrariamente grande ma G_perf = 0 — ovvero un simulatore fisicamente impreciso le cui policy ottimali siano comunque ottimali nel mondo reale. Questa separazione tra accuratezza del modello e ottimalità della policy è la motivazione fondamentale del framework bi-livello introdotto in 20.3.5. L'obiettivo del sim-to-real transfer non è costruire il simulatore fisicamente più fedele, bensì trovare la policy π^* che minimizza G_perf:

$$\pi^* = argmin_{\pi} G_{perf}(\mathcal{M}_s, \mathcal{M}_r, \pi) \quad (20.14)$$

20.3.2 Tassonomia sistematica delle fonti di divario

La survey di Aljalbout e colleghi (2025), destinata all'Annual Review of Control, Robotics, and Autonomous Systems, fornisce la classificazione più sistematica e aggiornata delle fonti di divario, articolando il problema in quattro dimensioni principali — dinamica, percezione, attuazione e progettazione del sistema — e identificando per ciascuna i sintomi che si manifestano al momento del trasferimento. La comprensione di questa tassonomia è il prerequisito per scegliere le strategie di mitigazione appropriate.

Il divario di dinamica (Gdyn) origina dalle semplificazioni che i simulatori adottano nella modellazione fisica. Le fonti principali includono: (i) l'assunzione di corpi rigidi, che ignora la deformabilità di link, giunti e oggetti manipolati; (ii) la modellazione approssimata dei contatti, dove simulatori come MuJoCo e Isaac Sim usano modelli di contatto morbido (spring-damper) o rigido con linearizzazione dei coni di attrito,

catturando solo parzialmente la dinamica di stick-slip e i cambi di stato di contatto; (iii) la parametrizzazione errata dei coefficienti fisici, come masse, inerzie, coefficienti d'attrito e costanti elastiche, la cui misurazione precisa è difficile e i cui valori cambiano nel tempo per usura; (iv) gli errori di integrazione numerica, che si accumulano su orizzonti temporali lunghi e dipendono dal metodo d'integrazione scelto (Eulero esplicito, Runge-Kutta, ecc.); (v) gli effetti non modellati, come la dinamica termica degli attuatori, la deriva delle sorgenti di alimentazione e l'usura meccanica.

Il divario di percezione (Gperc) misura la discrepanza tra i modelli di sensore simulati e quelli reali. Per le telecamere RGB, i renderer fotorealistici come quelli basati su ray-tracing omettono o approssimano effetti come l'aberrazione cromatica, le distorsioni della lente, il rolling shutter, i flare e la variabilità fine delle texture. Per i sensori di profondità, vengono ignorati effetti come il rumore dipendente dalla distanza, i pixel volanti nelle discontinuità di profondità e le interferenze da luce ambientale. Per i sensori LiDAR, la divergenza del fascio, la riflettività dipendente dal materiale e gli angoli di incidenza non vengono modellati accuratamente. In tutti i casi, il rumore reale è non-gaussiano, spazialmente correlato e dipendente dallo stato, mentre i simulatori tipicamente usano rumore bianco gaussiano a varianza fissa.

Il divario di attuazione ha origine nell'interfaccia tra la policy ad alto livello e i motori fisici. I simulatori trattano tipicamente gli attuatori come sistemi del primo ordine in grado di produrre coppie instantanee e lineari, mentre i motori reali esibiscono dinamiche di ordine superiore con costanti di tempo elettrica e meccanica non trascurabili, zone morte, backlash, saturazione coppia-velocità e isteresi. In presenza di riduttori ad alto rapporto (100:1 o superiore), questi effetti sono amplificati e il divario attuativo diventa dominante, come documentato da Kawachi (2025) nel contesto di robot bilancianti a due ruote con riduttori 110:1. I controllori di

basso livello — firmware proprietari con filtri anti-risonanza, limitatori di rate e logiche di protezione — introducono ulteriori latenze e non-linearità assenti nel simulatore.

20.3.3 Domain Randomization: formulazione, analisi formale e limiti

La domain randomization (DR), proposta da Tobin e colleghi (2017) e sistematizzata da Andrychowicz e colleghi (2020) nel contesto della manipolazione destra, rimane l'approccio più adottato nella pratica per affrontare il sim-to-real gap. L'idea fondamentale è addestrare la policy su una distribuzione di ambienti simulati ϱ(ξ) parametrizzata da un vettore ξ ∈ Ξ ⊂ ℝ^{d_ξ} che campiona diverse istanze dei parametri fisici e visuali del simulatore. L'obiettivo di addestramento diventa la massimizzazione del ritorno atteso su tutta la distribuzione:

$$\pi^{*} = argmax_{\pi} \mathbb{E}_{\xi \sim \rho(\xi)} \mathbb{E}_{\tau \sim \pi, \mathcal{M}(\xi)} \left[\sum_{t} \gamma^{t} r_{t} \right] \quad (20.15)$$

Il principio di copertura sottende la validità della DR: se la distribuzione ϱ(ξ) copre la realtà — ovvero se esiste ξ* ∈ Ξ tale che $\mathcal{M}(\xi^*) \approx \mathcal{M}_r$ — e se la policy π riesce a funzionare uniformemente bene per tutti i valori di ξ nel supporto di ϱ, allora sarà anche robusta nell'ambiente reale. Formalmente, posto che il gap di prestazione sia continuo rispetto a ξ e che ξ* appartenga al supporto di ϱ, si ha:

$$G_{perf}(\mathcal{M}_s, \mathcal{M}_r, \pi^{*}) \leq \varepsilon_{\rho} + \varepsilon_{DR} \quad (20.16)$$

dove ε_ϱ è l'errore di copertura (zero se ξ* è nel supporto di ϱ) e ε_DR è il gap di generalizzazione della policy sulla distribuzione ϱ. Questo risultato mostra che una DR sufficientemente ampia garantisce il trasferimento, ma al prezzo di un potenziale aumento dell'ε_DR: una distribuzione troppo ampia rende l'ottimizzazione difficile e la policy eccessivamente conservativa, poiché deve funzionare anche in ambienti fisicamente irrealistici.

La DR adattiva (ADR), proposta da OpenAI (Akkaya et al. 2019) nel contesto della manipolazione del cubo di Rubik, mitiga questo problema espandendo automaticamente i limiti di ξ in base alle prestazioni correnti della policy. Il meccanismo di ADR mantiene un curriculum ϱ_t(ξ) che viene aggiornato secondo una regola adattiva: se la policy riesce a risolvere il compito per una frazione di ambienti superiore a una soglia p_high, i limiti del parametro ξ vengono espansi; se la frazione scende sotto p_low, i limiti vengono ristetti. Formalmente:

$$\xi_{max}\ (t+1) = \xi_{max}(t) + \Delta \cdot \mathbb{1}\left[p(t) > p_{high}\right] \quad (20.17)$$

dove Δ è il passo di espansione. Il meccanismo complementare contrae il range quando p(t) < p_low. Questo curriculum garantisce che la policy sia sempre sfidana da ambienti al limite della propria competenza attuale, realizzando il concetto pedagogico della zona di sviluppo prossimale.

Kawachi (2025) ha identificato un limite fondamentale della DR pura che non riceve sufficiente attenzione nella letteratura: quando il gap tra simulatore e realtà è strutturale — dovuto ad esempio a riduttori ad alto rapporto o a dinamiche non lineari specifiche dell'hardware — l'amplificazione del range di randomizzazione non compensa la mancanza di accuratezza del modello, ma produce semplicemente policy che operano al minimo comune denominatore tra tutti i modelli randomizzati. Kawachi formalizza questa osservazione dimostrando che la policy DR ottimale converge verso la soluzione del problema di controllo pessimistico (minimax), il cui rendimento è upper-bounded dal rendimento nell'ambiente peggiore del supporto — che potrebbe essere molto lontano dall'ottimo reale. La proposta alternativa di Kawachi, discussa nella sezione 20.3.8, integra la DR con una regolarizzazione guidata dalle proprietà fisiche misurate direttamente sull'hardware, costringendo la rete a operare entro i limiti fisici effettivi del sistema reale.

20.3.4 Modelli residuali e adattamento del simulatore

Quando la DR non è sufficiente — perché il simulatore fa assunzioni fondamentalmente errate sulla fisica, come l'ipotesi di corpi rigidi per oggetti deformabili o il modello di contatto semplificato per dinamiche rich di contatto — un approccio alternativo e complementare è l'apprendimento di un modello residuale che corregge le predizioni del simulatore con un modello neurale addestrato su dati reali.

Formalmente, si definisce il bias del simulatore come la differenza tra la transizione reale e quella predetta dal modello simulato, lungo la stessa traiettoria stato-azione:

$$\Delta_t = s_{t+1}^{real} - f_\theta(s_t, a_t) \quad (20.18)$$

Se il bias Δ_t è stazionario o lentamente variante nel tempo, può essere approssimato da un modello residuale r_φ(s_t, a_t) ≈ Δ_t addestrato su un insieme di traiettorie reali τ^{real}_1, ..., τ^{real}_N mediante regressione supervisionata:

$$\varphi^* = argmin_\varphi \frac{1N}{\sum_{i=1}^{N}} \|r_\varphi(s_t, a_t) - \Delta_t\|^2 \quad (20.19)$$

Il modello corretto f_θ(s_t, a_t) + r_φ(s_t, a_t) può poi essere usato per la pianificazione o per generare dati di addestramento più realistici. Golemo et al. (2018) hanno per primi addestrato una rete LSTM sulle differenze tra traiettorie simulate e reali per migliorare il trasferimento. Ajay et al. hanno usato reti ricorrenti stocastiche per aumentare un simulatore deterministico con l'incertezza della dinamica di contatto. Hwangbo et al. (2019) hanno applicato questa strategia alla locomozione bipede, ottenendo il trasferimento zero-shot con soli pochi minuti di dati reali. Qi et al. (2023) l'hanno estesa alla manipolazione.

La limitazione principale del modello residuale è che cattura correttamente il gap solo nella distribuzione degli stati visitati durante la raccolta dei dati reali: in stati fuori distribuzione, il modello residuale

potrebbe produrre predizioni arbitrarie, introducendo un nuovo source di instabilità. Il framework bi-livello discusso nella sezione 20.3.5 affronta questa limitazione in modo più radicale, ottimizzando i parametri del simulatore direttamente rispetto alle prestazioni reali piuttosto che rispetto alla fedeltà delle predizioni.

20.3.5 Il framework bi-livello per l'ottimizzazione Sim2Real

Anand, Sawant, Hoffmann, Reinhardt e Gros (2025) hanno proposto un cambio di paradigma fondamentale nell'approccio al sim-to-real gap: invece di ottimizzare l'accuratezza del simulatore come proxy per le prestazioni reali, il framework bi-livello ottimizza i parametri del simulatore direttamente rispetto alle prestazioni della policy nell'ambiente reale. Questo approccio rompe l'assunzione implicita di tutti i metodi tradizionali — che un simulatore più fedele produca policy più performanti — e affronta il problema del disallineamento degli obiettivi (objective mismatch) che affligge l'apprendimento per rinforzo basato su modelli.

Il framework è formalizzato come un problema bi-livello: il livello interno risolve un problema di RL standard nel simulatore parametrizzato da θ, mentre il livello esterno adatta i parametri θ del simulatore per massimizzare le prestazioni della policy risultante nell'ambiente reale. Formalmente, il problema al livello esterno è:

$$\theta^{*} \in argmax_{\theta}\, \mathbb{E}_{\tau \sim \pi^{*}(\theta),\, P}\left[\sum_{t=0}^{\infty} \gamma^{t} r(s_t,\, a_t)\right] \; s.t. \;\; \pi^{*}(\theta) \in argmax_{\varphi}\, J(\pi\varphi,\, f_{\theta}) \quad (20.20)$$

dove P è il modello di transizione reale (sconosciuto ma sondabile mediante interazione), f_θ è il simulatore parametrizzato, e la notazione $\pi^{*}(\theta)$ indica che la policy ottimale del simulatore è una funzione implicita dei parametri θ. Il livello esterno non usa le transizioni simulate per aggiornarsi, bensì le transizioni reali raccolte durante l'esecuzione della policy nel mondo fisico.

La Sim2Real ottimalità è la condizione necessaria e sufficiente perché la policy ottimale nel simulatore sia anche ottimale nell'ambiente reale. Formalmente, l'MDP simulato con parametri θ raggiunge la Sim2Real ottimalità se e solo se:

$$argmax_a Q_\theta{}^*(s, a) = argmax_a Q^*(s, a), \ \forall s \in S \quad (20.21)$$

Questa condizione è più debole della corrispondenza esatta delle funzioni di valore: richiede solo che le azioni ottime coincidano, non che le funzioni Q abbiano lo stesso valore assoluto. Il punto cruciale è che la condizione (20.21) può essere soddisfatta anche da simulatori fisicamente inaccurati, purché il loro landscape di ottimizzazione guidi la policy verso le stesse soluzioni del problema reale.

Il gradiente rispetto ai parametri θ del ritorno reale è l'elemento computazionalmente critico del framework. Applicando la regola della derivata delle funzioni implicite alla dipendenza π*(θ), Anand et al. derivano l'espressione del policy gradient al livello esterno come una funzione delle sensitivities della policy rispetto ai parametri del simulatore:

$$\nabla_\theta J(\pi^*(\theta), P) = \mathbb{E}_{s,a \sim \pi^*(\theta)}\left[\nabla_\theta \pi^*(\theta)(a|s) \cdot A(s, a)\right] \quad (20.22)$$

dove A(s, a) = Q*(s, a) − V*(s) è la funzione di vantaggio nel mondo reale. Il termine ∇_θ π*(θ) misura come le azioni della policy ottimale del simulatore cambino al variare dei parametri θ, e si calcola mediante differenziazione implicita delle condizioni di ottimalità di Bellman all'interno del simulatore — un'operazione che richiede la differenziabilità del simulatore rispetto a θ, o l'uso di stimatori stocastici come lo score function estimator. La struttura bi-livello del framework è illustrata nella relazione tra il livello interno — RL standard nel simulatore — e il livello esterno — aggiornamento dei parametri θ sulla base del ritorno reale osservato.

L'import teorico di questo risultato va oltre la sua applicazione pratica: esso stabilisce che esiste un framework unificato in cui sim-to-real transfer e model-based RL in-the-loop condividono la stessa struttura bi-livello, e che la Sim2Real ottimalità è una proprietà dei parametri del simulatore, non della sua fedeltà fisica.

20.3.6 Digital twin dinamici: il paradigma Real-is-Sim

Abou-Chakra, Sun, Rana e colleghi (2025) hanno proposto un cambio di prospettiva altrettanto radicale con il framework Real-is-Sim: invece di costruire politiche che sopravvivono al trasferimento dal simulatore alla realtà, la policy opera sempre e solo sul simulatore, ma il simulatore è un digital twin che si sincronizza in tempo reale con il sistema fisico. Il robot reale non esegue le azioni della policy direttamente: si limita a inseguire gli stati giunti del robot simulato, che è il vero attuatore della policy.

Il componente centrale del framework è l'Embodied Gaussian Simulator, il primo simulatore a raggiungere la sincronizzazione in tempo reale con l'ambiente fisico in modo generalizzabile attraverso l'integrazione di Gaussian Splatting e fisica basata su particelle. A 60 Hz, le osservazioni visive e propriocettive del sistema reale correggono lo stato del simulatore, mantenendo il digital twin allineato con la realtà anche in presenza di perturbazioni o discrepanze dinamiche. Questo schema trasferisce la responsabilità del sim-to-real gap dalla policy al meccanismo di sincronizzazione del digital twin.

Formalmente, il loop di controllo Real-is-Sim è un sistema con due componenti: il simulatore S che evolve con la sua dinamica interna a partire dallo stato s^{sim}_t aggiornato dalle osservazioni reali z^{real}_t, e il robot reale che insegue lo stato simulato mediante un controllore di inseguimento ad alta banda:

$$s_{t+1}\text{^}\{sim\} = S\Big(s_t\text{^}\{sim\},\ \pi(s\_t\text{^}\{sim\}) + \alpha \cdot (z\_t\text{^}\{real\} - z\_t\text{^}\{sim\})\Big) \quad (20.23)$$

dove il termine di correzione con guadagno α mantiene il simulatore ancorato alle osservazioni reali, e $\pi(\cdot)$ è la policy addestrata interamente sulle rappresentazioni derivate dagli stati simulati. Il robot reale esegue poi la legge di inseguimento:

$$q_t^{real} \leftarrow q_t^{sim} \quad (20.24)$$

I risultati sperimentali di Real-is-Sim sul compito di manipolazione PushT mostrano che le valutazioni in ambiente virtuale sono coerenti con i risultati nel mondo reale, validando il digital twin come proxy affidabile per la valutazione delle policy. Il framework elimina la distinzione tra fase di addestramento e fase di deployment: il simulatore è sempre presente, anche durante l'esecuzione reale, fungendo da mediatore tra policy e robot.

20.3.7 Identificazione differenziabile dei parametri fisici: il framework HALO

Wang, Zhang, Xie e colleghi (2026) hanno proposto HALO (HeAvy-LOaded humanoid motion control), un framework che affronta il caso specifico del gap strutturale introdotto da carichi pesanti sulle braccia di robot umanoidi. La presenza di un payload sconosciuto altera sistematicamente la distribuzione di massa, il centro di massa (CoM) e la matrice di inerzia del sistema, introducendo un gap sim-to-real strutturale e di grande ampiezza che non può essere compensato dalla DR senza sacrificare l'agilità.

L'approccio di HALO è fondato sull'identificazione differenziabile dei parametri fisici: i parametri θ del modello del robot (masse, posizioni del CoM, inerzie) vengono ottimizzati direttamente dentro un simulatore differenziabile (MuJoCo XLA) minimizzando la discrepanza tra traiettorie simulate e traiettorie reali raccolte sull'hardware. La funzione obiettivo di identificazione è la somma di un termine di tracking e un termine di regolarizzazione:

$$L_{total}(\theta) = L_{track}(\theta) + \lambda R(\theta) \quad (20.25)$$

Il termine di tracking ℒ_track(θ) misura la discrepanza in spazio cartesiano tra le posizioni dei link del robot simulate dal modello θ e quelle reali, campionando frammenti di traiettoria di lunghezza N:

$$L_{track}(\theta) = L_{track}\text{^}\{all\}(\theta) + \alpha u\ L_{track}\text{^}\{upper\}(\theta) \quad (20.26)$$

$$L_{track}\text{^}\{all\}(\theta) = \frac{1}{B}\sum_{i=1}^{B}\frac{1}{N}\sum_{t=1}^{N}\sum_{j\in B}\|p_j\text{^}\{sim\}(t;\ \theta) - p_j\text{^}\{real\}(t)\|^{2} \quad (20.27)$$

dove B è l'insieme di tutti i link del corpo robotico, U ⊂ B è il sottoinsieme dei link dell'upper-body (più influenzati dal payload), α^u è un peso aggiuntivo per questi link, e B è il numero di frammenti di traiettoria campionati. La minimizzazione è eseguita con discesa del gradiente, sfruttando la differenziabilità di MuJoCo XLA per calcolare ∇_θ ℒ_track(θ) analiticamente:

$$\theta^{k+1} = \theta^{k} - \eta \cdot \nabla_{\theta} L_{total}(\theta\text{^}k) \quad (20.28)$$

La struttura a due stadi è la chiave della stabilità dell'identificazione: il primo stadio calibra l'intero modello del robot (senza payload) su traiettorie raccolte in condizioni scariche, correggendo le discrepanze intrinseche del modello nominale; il secondo stadio ottimizza solo i parametri legati al payload (massa, posizione CoM del carico) sulle traiettorie raccolte in condizioni caricate. Questa decomposizione previene il fenomeno dell'entanglement, per cui l'ottimizzazione attribuisce erroneamente al payload le discrepanze già presenti nel modello nominale. Con i parametri identificati, le policy di locomozione sono addestrate in simulazione e trasferite zero-shot all'hardware, ottenendo miglioramenti del 45-73% nella precisione del gait e del 73% nel salto yaw in condizioni di carico pesante.

20.3.8 Regolarizzazione guidata dalla fisica: guadagni proporzionali come surrogati

Kawachi (2025) propone un approccio originale e pragmatico alla chiusura del sim-to-real gap per piattaforme robotiche con riduttori ad alto rapporto: invece di identificare accuratamente l'intero modello fisico del sistema, che richiede sensori specializzati e procedure complesse, si misurano direttamente i guadagni proporzionali effettivi del sistema reale e si vincola la rete neurale a rispettarli durante l'addestramento. L'intuizione di fondo è che i guadagni proporzionali di un controllore PID — tradizionalmente trattati come parametri di design — possono essere reinterpretati come surrogati per l'insieme complesso delle dinamiche non modellate del sistema, incluse la larghezza di banda dell'attuatore, l'attrito e le dinamiche del riduttore.

Quattro guadagni proporzionali effettivi vengono misurati su esperimenti semplici nel mondo reale: k_x (posizione lineare), k_v (velocità lineare), k_θ (angolo), k_ω (velocità angolare). Questi guadagni caratterizzano la sensitività input-output del sistema fisico nelle quattro variabili di stato rilevanti. La regolarizzazione guidata dalla fisica è poi formulata come un termine di penalità nella funzione di loss dell'addestramento, che forza le sensitività input-output della rete neurale a coincidere con i guadagni misurati:

$$L_{phys}(\varphi) = \sum_{j\in\{x,v,\theta,\omega\}} \left(k_j\text{^}\{real\} - \left|\frac{\partial \pi_\varphi}{\partial z_j}\right| \right)^2 \quad (20.29)$$

dove π_φ è la rete neurale e ∂π_φ/∂z_j è la derivata parziale dell'output della rete rispetto alla variabile di stato j, valutata su un batch di stati raccolti in simulazione. La funzione di loss totale dell'addestramento è:

$$L_{total}(\varphi) = L_{RL}(\varphi) + \lambda L_{phys}(\varphi) \quad (20.30)$$

dove ℒ_RL è l'obiettivo di RL standard (es. la perdita del policy gradient) e λ bilancia i due termini. Il meccanismo di regolarizzazione impedisce al

simulatore di premiare guadagni virtuali irrealisticamente elevati — una delle cause principali del sim-to-real gap su riduttori ad alto rapporto — e mantiene il comportamento della policy all'interno dell'involucro di attuazione fisicamente raggiungibile.

La DR condizionata sui parametri dell'impianto completa il framework: i parametri fisici correnti dell'impianto (guadagni, inerzie, ecc.) sono aggiunti come input alla policy, consentendo alla rete di specializzare il proprio comportamento per il particolare esemplare hardware invece di dover computare la soluzione di compromesso che funziona per tutti i possibili valori randomizzati. Questo approccio, combinato con la regolarizzazione guidata dalla fisica, ha consentito di raggiungere su un robot bilanciante a due ruote con riduttore 110:1 tempi di assestamento angolare in hardware molto prossimi a quelli simulati, mentre una policy DR pura mostrava oscillazioni persistenti.

20.3.9 Imitazione del MPC e trasferimento sim-to-real nel controllo ottimo

Kim, Lee, Han e Choi (2026) affrontano il sim-to-real gap da una prospettiva di controllo piuttosto che di apprendimento, ispirandosi alla teoria del Robust Tube Model Predictive Control (Tube MPC). Il contesto è quello dell'imitazione dell'MPC mediante reti neurali profonde (Deep Neural Networks, DNN): l'MPC è un controllore computazionalmente costoso che risolve un problema di ottimizzazione in tempo reale, e l'obiettivo dell'imitation learning è sostituirlo con una DNN che produca azioni simili con un costo computazionale molto inferiore (riduzione di 10-20x). Tuttavia, l'addestramento dell'imitatore in simulazione introduce un sim-to-real gap: in stati reali che differiscono da quelli simulati, la DNN si trova fuori dalla propria distribuzione di addestramento (out-of-distribution, OOD) e può produrre azioni errate.

Il framework proposto è basato sull'invariante di tubo: si definisce un tubo invariante nel spazio degli stati attorno alla traiettoria nominale del sistema, e si garantisce che la DNN operi sempre all'interno di questo tubo progettando un controllore ausiliario che corregge le deviazioni causate dal gap. Formalmente, sia e_t = x_t - x_t^{nom} la deviazione dello stato reale rispetto alla traiettoria nominale. Il controllore ausiliario κ(e_t) è progettato per portare e_t verso zero, mantenendo lo stato all'interno del tubo Ω = {e : |e| ≤ r}:

$$u_t = u_{DNN}(x_t\text{^}\{nom\}) + \kappa(e_t) \quad (20.31)$$

Il controllore κ(e) è derivato dalla teoria del Tube MPC robusto: si definisce la matrice K = argmin_{K} J_{LQR}(A+BK, Q, R) soluzione del problema LQR per le dinamiche linearizzate del sistema, e si pone κ(e) = Ke. La regione invariante Ω è allora il più piccolo insieme invariante S con rispetto al sistema lineare e_t+1 = (A+BK)e_t + w, dove w è il disturbo dovuto al gap. Il teorema fondamentale del Tube MPC garantisce che se e_0 ∈ Ω, allora e_t ∈ Ω per ogni t > 0, ovvero lo stato reale non si discosta mai dalla traiettoria nominale oltre una distanza controllata dal gap atteso:

$$e_0 \in \Omega \Rightarrow e_t \in \Omega, \ \forall t \geq 0 \quad (20.32)$$

Intuitivamente, il controllore ausiliario Ke funge da ammortizzatore rispetto alle perturbazioni causate dal gap tra simulatore e realtà: quando la DNN opera su uno stato che nella realtà è leggermente diverso da quello simulato, il contributo Ke riporta il sistema all'interno della bolla di fiducia della DNN. Questo garantisce che la DNN non si trovi mai in stati troppo lontani dalla propria distribuzione di addestramento. Il framework include anche un Input Refinement Governor che corregge le azioni della DNN quando i parametri del modello variano rispetto a quelli usati nell'addestramento, consentendo alla DNN di soddisfare i vincoli dell'MPC anche in condizioni di incertezza parametrica. Validato su

cart-pole e controllo della traiettoria di veicoli, il framework riduce le violazioni dei vincoli di quasi l'80% rispetto alla DNN priva del tubo.

20.3.10 Manipolazione destra: il gap tattile, attuativo e di contatto

Zhao, Dong, He e colleghi (ByteDance Seed, 2026) hanno sviluppato il primo sistema di RL sim-to-real per la manipolazione destra force-sensitive con deployment zero-shot su una mano a cinque dita e dodici gradi di libertà. Il lavoro è significativo perché identifica e affronta sistematicamente i due gap specifici della manipolazione destra che ne hanno impedito finora la diffusione: il gap di simulazione tattile e il gap di attuazione.

Il gap di simulazione tattile origina dalla complessità computazionale della simulazione ad alta risoluzione dei sensori tattili: le librerie esistenti basate su FEM (elementi finiti) o su rendering fisico di elastomeri sono troppo lente per il ciclo di addestramento RL, che richiede milioni di passi di simulazione. La proposta di Zhao et al. è un approccio di approssimazione geometrica: la simulazione tattile è implementata come calcolo delle distanze tra un'array densa di unità tattili virtuali e la superficie dell'oggetto, sfruttando la cinematica diretta parallela:

$$c_i = ReLU\left(r - d_i\left(q_t, x_{obj}\right)\right) \quad (20.33)$$

dove c_i è la lettura del sensore tattile all'unità i, d_i(q_t, x_obj) è la distanza tra l'unità i e la superficie dell'oggetto calcolata mediante cinematica diretta dalla configurazione q_t e dalla posa dell'oggetto x_obj, e r è il raggio di attivazione tattile. Questa formulazione consente il calcolo parallelo su GPU dell'intera distribuzione di segnali tattili in modo differenziabile, senza richiedere simulazione FEM.

Il gap attuativo origina dalla mancanza di sensori di coppia diretti sulle dita — la maggior parte delle mani destre commerciali usa la corrente del motore come proxy della coppia, ma la relazione corrente-coppia è non

lineare e fortemente dipendente dalla velocità angolare per le dinamiche dei rotori. La calibrazione corrente-coppia è modellata come:

$$\tau_j = k_\tau(\theta_j) \cdot i_j - b_j(\theta_j) \quad (20.34)$$

dove τ_j è la coppia stimata al giunto j, i_j è la corrente misurata, θ_j è la velocità angolare del giunto, k_τ(θ_j) è il guadagno coppia-corrente dipendente dalla velocità (curva K_T), e b_j(θ_j) è il termine di bias che cattura l'attrito viscoso e il back-EMF. La randomizzazione delle dinamiche dell'attuatore — saturazione coppia-velocità, isteresi, backlash — è poi aggiunta come DR sull'attuatore simulato per coprire le variabilità inter-giunto e inter-esemplare. Con questo approccio, l'actor-critic asimmetrico PPO addestrato interamente in simulazione trasferisce zero-shot su hardware reale le skill di inseguimento della forza di presa e di riorientazione in-hand dell'oggetto, dimostrando che il sim-to-real RL è applicabile alla manipolazione destra force-sensitive quando i gap tattile e attuativo sono affrontati sistematicamente.

20.3.11 Informazione privilegiata, adattamento rapido e apprendimento della rappresentazione

Una strategia trasversale e particolarmente efficace per il sim-to-real transfer è l'uso dell'informazione privilegiata (privileged information): dati disponibili durante l'addestramento in simulazione ma non durante il deployment reale, come i parametri fisici esatti del simulatore, la posa precisa degli oggetti, le forze di contatto. Pinto e colleghi (2018) hanno proposto l'architettura actor-critic asimmetrica, in cui il critic è condizionato sull'informazione privilegiata durante l'addestramento (producendo un segnale di valore più informativo), mentre l'actor vede solo le osservazioni disponibili a deployment. Formalmente:

$$V(s, \chi) \approx V_\omega(o, \chi); \quad \pi(\cdot|s) \approx \pi_\varphi(\cdot|o) \quad (20.35)$$

dove χ è l'informazione privilegiata disponibile solo al critic, o è l'osservazione reale disponibile anche all'actor, e s è lo stato completo. L'actor impara a massimizzare il ritorno usando solo le osservazioni che avrà a disposizione nel mondo reale, ma guidato da un critic che conosce la fisica esatta — un analogo dell'insegnamento socratico in cui il tutor (critic) conosce la risposta ma il discente (actor) deve ricavarla dalle sole percezioni.

Il Rapid Motor Adaptation (RMA, Kumar et al. 2021) porta questo principio un passo oltre: invece di ignorare l'informazione privilegiata durante il deployment, il sistema impara a inferirla in tempo reale dalle osservazioni recenti. RMA divide il problema in due fasi: nella prima, una policy priviliged π_p(a | o, z) è addestrata con il vettore z di parametri fisici privilegiati come input addizionale; nella seconda, un encoder adattivo φ(o_{t-k:t}) è addestrato per inferire una rappresentazione latente $\hat{z} \approx z$ dalle osservazioni recenti:

$$\varphi^{*} = argmin_{\varphi}\, \mathbb{E}_{o\_\{t-k:t\}}\left[\left\|\varphi(o_{t-k:t}) - z\right\|^{2}\right] \quad (20.36)$$

Al deployment, la policy usa π_p(a | o, φ(o_{t-k:t})) adattando implicitamente il comportamento ai parametri fisici inferiti dall'adattamento motorio recente. Questo consente alla policy di rispondere a variazioni dei parametri fisici che non possono essere anticipate durante l'addestramento, incluse usura degli attuatori, variazioni di terreno e payload sconosciuti.

La quantificazione bayesiana dell'incertezza offre un framework complementare per la pianificazione robusta rispetto al gap: il world model mantiene una distribuzione a posteriori p(θ|D) sui propri parametri, e pianifica in modo worst-case rispetto a essa:

$$a^{*} = argmax_{a\_\{0:H\}}\, min_{\theta \sim p(\theta|D)}\, \mathbb{E}_{\theta}\left[\sum_{t=0}^{H} \gamma^{t} r(s, a)\right] \quad (20.37)$$

Questo obiettivo minimax corrisponde a una strategia di pianificazione pessimistica: l'agente sceglie le azioni che massimizzano il ritorno nel caso peggiore in termini di parametri del modello. In pratica, la distribuzione a posteriori $p(\theta|D)$ è approssimata con Deep Ensemble (Lakshminarayanan et al. 2017) o con processi gaussiani variazionali, e il minimax è approssimato con campionamento.

20.3.12 Sintesi: una gerarchia di strategie e la scelta appropriata

Le strategie per il sim-to-real transfer si organizzano in una gerarchia di complessità crescente, dove ciascun livello si aggiunge al precedente quando le strategie più semplici non sono sufficienti.

Al livello base, la scelta delle modalità di osservazione e azione influenza fortemente l'entità del gap: rappresentazioni come le point cloud di profondità hanno un gap percettivo minore delle immagini RGB; azioni nello spazio delle velocità articolari hanno un gap attuativo minore delle azioni nello spazio delle coppie; la regolarizzazione delle azioni (penalizzare variazioni brusche) riduce il gap di controllo. Al livello successivo, la domain randomization copre il gap introducendo variabilità artificiale nelle distribuzioni dei parametri; la DR adattiva (ADR) automatizza la scelta del range. Quando il gap strutturale non è coperto dalla DR, si ricorre a: (i) modelli residuali per correggere le predizioni del simulatore con pochi dati reali; (ii) identificazione differenziabile dei parametri per calibrare il modello fisico prima dell'addestramento; (iii) regolarizzazione guidata dalla fisica per vincolare la rete alla fisica misurabile; (iv) framework bi-livello per ottimizzare i parametri del simulatore direttamente rispetto alle prestazioni reali. Al livello più alto, i digital twin dinamici (Real-is-Sim) eliminano la distinzione tra simulazione e realtà mantenendo il simulatore sincronizzato in tempo reale con il sistema fisico, trasferendo il burden del gap dalla policy al meccanismo di sincronizzazione.

Nessuna di queste strategie è universalmente superiore: la scelta dipende dalla disponibilità di dati reali, dal tipo di gap dominante, dalle risorse computazionali e dalla tolleranza al rischio. Le soluzioni più robuste combinano più livelli: ad esempio, un sistema tipico di locomozione biped combina identificazione dei parametri del simulatore, DR sui parametri residui, actor-critic asimmetrico con informazione privilegiata, e RMA per l'adattamento rapido in deployment.

26.4 Connessioni Trasversali: Causalità, Composizionalità e Trasferimento

Le tre sfide aperte analizzate nel Capitolo 26 — causalità, composizionalità e divario simulazione-realtà — non sono problemi indipendenti: presentano connessioni strutturali profonde che suggeriscono approcci integrati.

La connessione tra causalità e sim-to-real gap è particolarmente diretta: il gap di dinamica origina precisamente dalla modellazione errata delle relazioni causali tra azioni e stati successivi. Un world model causale — che usa lo Structural Causal Model (SCM) per rappresentare le dinamiche invece di un modello puramente correlazionale — dovrebbe in linea di principio essere più robusto al sim-to-real gap, poiché le relazioni causali tra azioni e stati tendono a essere più stabili tra domini rispetto alle correlazioni statistiche osservazionali. Questa intuizione è coerente con la Causal Representation Learning (CRL): le rappresentazioni latenti causalmente disentangled, in cui ciascuna coordinata corrisponde a una variabile causale del meccanismo generativo, dovrebbero trasferirsi meglio tra simulazione e realtà rispetto alle rappresentazioni distribuite standard.

La connessione tra composizionalità e sim-to-real gap emerge nei compiti di manipolazione di oggetti nuovi: un sistema con rappresentazioni oggetto-centriche che conosce le proprietà fisiche individuali degli oggetti

(massa, coefficiente di attrito, deformabilità) può in linea di principio costruire predizioni delle dinamiche di contatto componendo le rappresentazioni dei singoli oggetti, piuttosto che affidarsi a un modello monolitico che può fallire su oggetti mai visti in simulazione. Gli Object-Centric World Model (OCWM) discussi nella sezione 20.2 rappresentano la prima generazione di architetture che mirano a questa proprietà composizionale nel contesto dei world model fisici.

Il ciclo chiuso tra queste tre sfide suggerisce che la soluzione a lungo termine al sim-to-real gap non è puramente tecnica — costruire simulatori sempre più fedeli — ma piuttosto epistemica: costruire rappresentazioni del mondo che siano causalmente corrette, compositivamente organizzate e trasferibili per struttura piuttosto che per distribuzione. Questa visione converge con l'agenda della Causal Representation Learning e con il framework JEPA gerarchico discusso nel Capitolo 24, e costituisce uno dei programmi di ricerca più ambiziosi e promettenti della robotica cognitiva contemporanea.

PARTE V

Robotica Formale e Applicativa

Capitolo 27 — Pianificazione del Moto su Varietà e Spazi di Configurazione

La pianificazione del moto è il problema di determinare una traiettoria eseguibile che porti un sistema robotico da una configurazione iniziale a una configurazione obiettivo evitando ostacoli e rispettando vincoli cinematici e dinamici. Il fondamento matematico di questo problema è la topologia dello spazio delle configurazioni, introdotta da Lozano-Pérez (1983) e sistematizzata da Latombe (1991). Comprendere la struttura di tale spazio è necessario non solo per progettare algoritmi efficienti, ma anche per inquadrare correttamente i world model latenti discussi nella Parte III: un world model che operi direttamente nello spazio delle configurazioni può pianificare traiettorie senza ricorrere alla rappresentazione esplicita degli ostacoli nello spazio cartesiano.

27.1 Topologia dello Spazio delle Configurazioni

Dato un robot con n gradi di libertà, lo spazio delle configurazioni C è la varietà differenziabile i cui punti corrispondono biunivocamente alle configurazioni del robot. Per un manipolatore rotoidale a n giunti revoluti, C = T^n (il toro n-dimensionale), dove ogni coordinata q_i ∈ S^1 rappresenta l'angolo del giunto i-esimo. La presenza di ostacoli nello spazio fisico induce un sottoinsieme C_obs ⊂ C detto spazio di configurazioni ostacolo, definito come l'immagine inversa della mappa di configurazione degli ostacoli. Lo spazio libero C_free = C \ C_obs è la varietà su cui si cerca la traiettoria pianificata.

$$C\_obs = \{\, q \in C : A(q) \cap B \neq \varnothing \,\} \quad (27.1)$$

dove A(q) è la regione occupata dal robot nella configurazione q e B è la regione occupata dagli ostacoli. La struttura topologica di C_free dipende in modo non banale dalla geometria degli ostacoli e dall'architettura cinematica del robot. Per manipolatori planari con ostacoli poligonali, C_free può essere semialgebrica; per robot che operano in ambienti tridimensionali, C_free è generalmente un aperto di

una varietà di Lie (si veda il Capitolo 2 per la struttura di SO(3) e SE(3)). La pianificazione del moto si riduce al problema di trovare un cammino continuo $\gamma : [0,1] \to C\_free$ tale che $\gamma(0) = q\_start$ e $\gamma(1) = q\_goal$.

La completezza di un algoritmo di pianificazione è la garanzia che, se una soluzione esiste, l'algoritmo la trova in tempo finito. La completezza risoluta, dimostrata per algoritmi basati su rappresentazione cellulare esplicita (Canny 1988), ha costo computazionale esponenziale in n (O(n^2n)), che la rende inapplicabile per robot con molti gradi di libertà. Gli algoritmi probabilisticamente completi, che convergono alla soluzione con probabilità 1 al crescere del numero di campioni, rappresentano il compromesso pratico adottato dalla quasi totalità dei sistemi robotici moderni.

27.2 Campionatori Probabilistici: RRT e RRT*

Il Rapidly-exploring Random Tree (RRT), introdotto da LaValle (1998), costruisce incrementalmente un albero T radicato in q_start espandendosi verso configurazioni campionate casualmente in C. L'algoritmo fondamentale itera tre passi: campionamento di $q\_rand \in C$, selezione del nodo q_near più vicino nell'albero, estensione verso q_rand di un passo δ per ottenere q_new, e inserimento di q_new in T se il segmento $[q\_near, q\_new] \subset C\_free$. La proprietà di rapidly-exploring deriva dalla distribuzione di Voronoi: la probabilità che un nodo venga selezionato come q_near è proporzionale al volume della sua cella di Voronoi, il che spinge l'albero verso le regioni inesplorate.

$$q\_new = q\_near + \delta \cdot (q\_rand - q\_near) / \|q\_rand - q\_near\| \quad (27.2)$$

RRT è probabilisticamente completo ma non ottimale: il cammino trovato converge a una soluzione sub-ottimale al crescere dei campioni. RRT* (Karaman e Frazzoli 2011) risolve il problema di ottimalità aggiungendo due operazioni: la selezione del genitore ottimale (choose parent) e il ricablaggio dell'albero (rewire). Nella scelta del genitore, q_new viene connesso al nodo $q\_best \in B(q\_new, r)$ che minimizza il costo del cammino dalla radice, dove $B(q\_new, r)$ è la sfera di raggio $r = \gamma(\log n / n)^{1/d}$ intorno a q_new (con d = dim(C) e n numero di nodi correnti). Il raggio r è scelto in modo che il numero di vicini nell'intorno cresca logaritmicamente, garantendo sia la convergenza che l'efficienza computazionale.

$$c^*(q\_new) = \min_{q \in B(q\_new,r) \cap T} [\, cost(q) + d(q, q\_new) \,], \ \text{se } [q, q\_new] \subset C\_free \quad (27.3)$$

Il teorema principale di Karaman e Frazzoli (2011) stabilisce che RRT* è asintoticamente ottimale: con probabilità 1, il costo del cammino trovato converge al costo ottimale c* al crescere del numero di campioni n → ∞. La velocità di convergenza è O(log n / n)^{1/d}, il che implica che per spazi di dimensione elevata (d ≥ 10, come in molti robot umanoidi) la convergenza pratica richiede un numero molto grande di campioni. Varianti come Informed RRT* (Gammell et al. 2014) restringono il campionamento all'ellissoide prolato che contiene tutte le soluzioni potenzialmente migliori della soluzione corrente, accelerando la convergenza di un fattore proporzionale al rapporto tra il volume dell'ellissoide e quello di C.

27.3 Mappe Probabilistiche di Configurazione: PRM

La Probabilistic Roadmap (PRM), introdotta da Kavraki et al. (1996), costruisce offline una mappa del C_free campionando casualmente configurazioni libere e collegandole con un planner locale (tipicamente una connessione in linea retta in C). L'algoritmo si divide in una fase di costruzione e una fase di interrogazione. Nella costruzione, si campionano N configurazioni libere q_1, ..., q_N e si costruisce il grafo G = (V, E) dove V = {q_i} e un arco (q_i, q_j) ∈ E se e solo se d(q_i, q_j) ≤ r e il segmento q_iq_j ⊂ C_free. La PRM è particolarmente adatta a ambienti statici con molte query di pianificazione, poiché la costosa fase di costruzione viene ammortizzata su tutte le query successive.

La teoria dei grafi garantisce che la PRM sia probabilisticamente completa se il numero di campioni N e il raggio r soddisfano r ≥ γ(log N / N)^{1/d} (analogo alla condizione di RRT*). In pratica, la scelta di r dipende dalla geometria di C_free: ambienti con passaggi stretti richiedono campionamento adattativo concentrato nei bottleneck, realizzato da strategie come bridge testing (campionamento nei punti medi tra configurazioni in collisione) o Gaussian sampling. La connessione tra PRM e world model latenti emerge nell'apprendimento di grasp e manipolazione: un encoder che mappa le osservazioni in un latent space dove le metriche di distanza approssimano la metrica geodesica su C_free costituisce una PRM implicita apprendibile, realizzata ad esempio da Universal Value Function Approximators (UVFA) e da latent-space planning (Hafner et al. 2020).

27.4 Pianificazione di Traiettorie Ottimali: CHOMP e TrajOpt

I metodi basati su campionamento (RRT*, PRM) trovano cammini geometricamente fattibili ma non ottimizzano esplicitamente criteri di qualità cinematica o dinamica. I metodi di ottimizzazione di traiettoria formulano la pianificazione come un problema variazionale: trovare la traiettoria $\xi = \{q_t\}_{t=0}^{T}$ che minimizza un funzionale di costo $C[\xi] = C_{smooth}[\xi] + \lambda\, C_{obs}[\xi]$, dove C_{smooth} penalizza le irregolarità della traiettoria e C_{obs} penalizza la vicinanza agli ostacoli.

CHOMP (Covariant Hamiltonian Optimization for Motion Planning, Ratliff et al. 2009) formula C_{smooth} come l'energia cinetica media della traiettoria:

$$C_{smooth}[\xi] = \tfrac{1}{2} \int_0^T \|\dot{s}_t\|^2 \, dt = \tfrac{1}{2}\, \xi^T A\, \xi \quad (27.4)$$

dove A è la matrice differenziale del secondo ordine discreta ($A = K^T K$, con K l'operatore di differenza finita sulla traiettoria discretizzata). Il costo di ostacolo è:

$$C_{obs}[\xi] = \int_0^T \sum_{i=1}^{n_b} c(x_i(\xi_t)) \, \|\dot{x}_i(\xi_t)\| \, dt \quad (27.5)$$

dove $x_i(\xi_t)$ è la posizione del corpo i in configurazione q_t, $\dot{x}_i$ è la sua velocità cartesiana, e $c(\cdot)$ è un potenziale di costo basato sulla signed distance function (SDF) del campo di ostacoli. L'aggiornamento di CHOMP è un gradiente funzionale covariante (Riemanniano rispetto alla metrica A):

$$\xi_{k+1} = \xi_k - \eta\, A^{-1} (\alpha\, A\, \xi_k + \nabla_\xi C_{obs}[\xi_k]) \quad (27.6)$$

La formulazione covariante garantisce che l'aggiornamento sia invariante rispetto alla parametrizzazione temporale della traiettoria, evitando la lentezza di convergenza dei gradienti nello spazio euclideo. TrajOpt (Schulman et al. 2014) estende questo approccio sostituendo il gradiente continuo con vincoli di non-collisione formulati come distanze convesse (via convex-concave procedure, CCCP), permettendo di gestire anche vincoli di cinematica inversa e di torque limite. La combinazione di RRT* (per trovare una soluzione iniziale globalmente fattibile) con CHOMP o TrajOpt (per ottimizzarla localmente) è la strategia standard dei sistemi di manipolazione di livello industriale.

27.5 Motion Planning Differenziabile e World Models Latenti

La connessione tra pianificazione del moto e world model latenti si manifesta più chiaramente nel paradigma del motion planning differenziabile su SE(3). Un world

model $\varphi : O \to Z$ che mappa osservazioni $o \in O$ in rappresentazioni latenti $z \in Z$ può essere interpretato come un encoder implicito dello spazio delle configurazioni: se φ è addestrato a predire le conseguenze future delle azioni, la metrica indotta nel latente approssima la metrica di raggiungibilità in C_free. Il planning può quindi essere eseguito direttamente nel latente tramite ottimizzazione:

$$z_{0:T}^* = \arg\min_{z_{0:T}} \sum_t C(z_t, a_t) + \lambda\, D(z_T, z_{goal}) \quad (27.7)$$

dove $D(\cdot, \cdot)$ è una metrica di distanza nel latente e $z_{goal} = \varphi(o_{goal})$ è la rappresentazione latente della configurazione obiettivo. Questo approccio è adottato da sistemi come DreamerV3 (Hafner et al. 2023) per il planning in loop chiuso, e da MBRL (Model-Based Reinforcement Learning) con MPC neurale (si veda il Capitolo 24 sulla JEPA e la robotica embodied per il collegamento con il framework LPA). La differenziabilità del world model permette di backpropagare il gradiente del costo rispetto alla traiettoria latente, realizzando un CHOMP nel latente invece che in C. La limitazione principale è che le garanzie di completezza e ottimalità di RRT* o CHOMP non si trasferiscono automaticamente al planning latente senza assunzioni sulla struttura metrica di Z.

Capitolo 28 — Grasping, Manipolazione con Contatto e Controllo di Impedenza

Il problema della presa (grasp) è il nucleo della manipolazione robotica: determinare le forze di contatto che un insieme di dita o di superfici di presa può esercitare su un oggetto in modo da stabilizzarlo contro perturbazioni esterne. Nonostante la sua apparente semplicità pratica, il problema della presa è matematicamente profondo: coinvolge la meccanica del contatto con attrito, la geometria differenziale delle superfici di contatto, e l'ottimizzazione su spazi non convessi. Il controllo di impedenza, che governa la relazione forza-posizione all'interfaccia robot-ambiente, fornisce il formalismo dinamico per eseguire una presa pianificata in presenza di incertezze geometriche e dinamiche.

28.1 Meccanica del Contatto e Modello di Attrito di Coulomb

Il modello di Coulomb per il contatto con attrito secco è il punto di partenza standard per l'analisi delle forze di contatto in robotica. Una forza di contatto f ∈ R^3 applicata in un punto di contatto p con normale di contatto ën può essere decomposta in componente normale f_n = (f·ën)ën e componente tangenziale f_t = f - f_n. Il modello di Coulomb stabilisce che la forza è fisicamente realizzabile se e solo se appartiene al cono di attrito associato al coefficiente di attrito μ:

$$FC(\mu) = \{ f \in R^3 : \|f_t\| \leq \mu f_n, f_n \geq 0 \} \quad (28.1)$$

Il cono di attrito FC(μ) è un cono convesso in R^3 con asse allineato alla normale di contatto e semi-angolo α = arctan(μ). La linearizzazione del cono di attrito produce il cono friction cone approssimato (FC_k) come inviluppo positivo di k generatori u_1, ..., u_k equidistribuiti sull'ellisse di base: f ∈ FC_k se f = β_0 ën + ∑_{i=1}^k β_i u_i, con β_i ≥ 0. La linearizzazione trasforma il problema di ottimizzazione con vincoli di secondo ordine (cone) in un problema lineare (LP), che è alla base della maggior parte degli algoritmi di sintesi di grasp.

28.2 Force Closure e Metriche di Qualità della Presa

Una presa è in force closure se le forze di contatto applicabili all'interno dei coni di attrito possono generare qualsiasi resultante wrench (forza e momento) nello spazio dei wrench W = R^6. Formalmente, una presa con m punti di contatto con posizioni p_i e normali ën_i è in force closure se il cono positivo generato dalle colonne di G^T (dove G è la matrice di grasp) è uguale all'intero spazio R^6:

$$FC\text{-}grasp: cone\{ G^T (u_1, ..., u_{mk})^T \} = R^6 \quad (28.2)$$

La matrice di grasp G ∈ R^{6×m} mappa le forze di contatto nello spazio dei wrench: la i-esima colonna di G^T è $[f_i^T, (p_i \times f_i)^T]^T$, dove p_i × f_i contribuisce al momento. Il volume dell'ellissoide inscritto nel convex hull di $G^T u_i$ (noto come epsilon-quality metric, Salisbury e Roth 1983) fornisce una misura scalare della qualità della presa: valori maggiori indicano una presa più robusta a perturbazioni esterne. La qualità ε = min_{w : ‖w‖=1} max_{f ≥ 0, Gf=w} 1 equivale al raggio della più piccola sfera inscrivibile nell'inviluppo convesso dei wrench di contatto.

Le reti neurali per la sintesi di grasp (GraspNet-1Billion, Fang et al. 2020; AnyGrasp, Fang et al. 2023) apprendono implicitamente una proxy della qualità della presa da milioni di grasp annotati tramite simulazione fisicamente basata. GraspNet-1Billion

addestra una rete che, data la nuvola di punti della scena, predice per ogni punto un set di pose di presa candidate con punteggi di qualità. AnyGrasp estende questo approccio con una rete Transformer che integra informazioni di profondità, colore e contesto semantico, raggiungendo generalizzazione zero-shot su oggetti mai visti. La connessione con i world model latenti della Parte III è diretta: l'encoder di percezione di AnyGrasp produce una rappresentazione latente della scena che funge da world model approssimato per la pianificazione della presa.

28.3 Controllo di Impedenza: Fondamenti Porto-Hamiltoniani

Il controllo di impedenza, introdotto da Hogan (1985), governa la relazione dinamica tra la forza F_ext applicata all'end-effector e la deviazione x_e = x - x_d rispetto alla posizione desiderata x_d. L'impedenza meccanica desiderata è modellata come un sistema massa-molla-smorzatore di secondo ordine:

$$M_d\,\ddot{x}_e + D_d\,\dot{x}_e + K_d\,x_e = F_{ext} \quad (28.3)$$

dove M_d ∈ R^{6×6}, D_d ∈ R^{6×6}, K_d ∈ R^{6×6} sono rispettivamente le matrici di inerzia, smorzamento e rigidezza desiderate. Selezionando M_d = M(q) (inerzia del robot in configurazione q), il legge di controllo che realizza questa impedenza nello spazio cartesiano è:

$$\tau = J^{T}(q)\,[\,M(q)\,(J^{N}\ddot{x}_d - \dot{C}(q,\dot{p})\dot{p}) - D_d\,\dot{x}_e - K_d\,x_e + F_{ext}\,] + C(q,\dot{p})\dot{p} + g(q) \quad (28.4)$$

Il fondamento porto-hamiltoniano del controllo di impedenza emerge considerando la struttura della dinamica del robot come sistema a porte (port-Hamiltonian system). Definendo l'Hamiltoniano H(q,p) = ½p^T M^{-1}(q)p + V(q) (energia totale del robot) e le porte di interazione (f_p, e_p) = (ṗ, τ) (velocità e coppia generalizzate), la dinamica hamiltoniana è ẋ = (J - R)∇H + g u, dove J è la matrice simmetrica antisimmetrica e R ≥ 0 è la matrice di dissipazione. Il controllo di impedenza agisce sulla porta di dissipazione: la matrice D_d scelta corrisponde all'aggiunta di un termine di smorzamento virtuale che è intrinsecamente passivo (R_d = D_d ≥ 0), garantendo la stabilità dell'interazione con ambienti passivi per qualsiasi rigidezza ambientale positiva. Questa proprietà di passività garantita è il vantaggio principale del controllo di impedenza rispetto ai controllori puramente in posizione o in forza, che possono diventare instabili in contatto con ambienti rigidi.

28.4 Controllo Ibrido Posizione-Forza per Task di Assemblaggio

Il controllo ibrido posizione-forza (Raibert e Craig 1981) decompone lo spazio dei compiti in una sottospazio di posizione Ω_p, dove il contatto non impone vincoli di forza, e un sottospazio di forza Ω_f, dove i vincoli di contatto impongono la forza. La selezione dei sottospazi è realizzata da una matrice di selezione S ∈ R^{6×6} (con S^2 = S, SS^T = S) che proietta i comandi nello spazio appropriato: il controllore in posizione agisce su S x e il controllore in forza su (I - S) f.

Nei task di assemblaggio (peg-in-hole, bearing insertion, connector mating), i vincoli di contatto impongono che la forza normale sia non nulla (il componente deve essere in contatto con la guida) mentre la posizione tangenziale deve essere regolata per centrare il peg nel foro. La difficoltà principale è la stima in tempo reale della matrice di selezione S, che dipende dalla geometria di contatto istantanea. I sistemi moderni come ForceBased Reactive Manipulation (FBRM) e TaskForce (Luo et al. 2024) sostituiscono la selezione esplicita con politiche apprese via RL che regolano implicitamente il bilanciamento forza-posizione in base all'osservazione delle forze al polso tramite sensori F/T. La connessione con i world model emerge nell'idea di world model di forza: un modello che predice le forze di reazione attese date la traiettoria pianificata e le proprietà del materiale può guidare il controllore ibrido adattando S dinamicamente.

28.5 Grasp con Reti Neurali Profonde: da GraspNet a AnyGrasp

La pipeline standard per la sintesi di grasp con reti profonde consiste di tre fasi: percezione della scena (nuvola di punti o immagine di profondità), generazione di candidati di presa, e scoring/selezione della presa ottimale. GraspNet-1Billion (Fang et al. 2020) realizza le ultime due fasi con un'architettura in due stadi: un backbone di percezione 3D (PointNet++ o SparseConv) produce feature per punto, e un modulo di generazione produce per ogni punto di ancoraggio un insieme di pose di presa (orientamento del gripper) con punteggi di qualità φ(p, θ) ∈ [0,1].

Il training supervisiona il punteggio φ con etichette derivate dalla simulazione fisica: una presa (p, R_g) è positiva se la simulazione MuJoCo con perturbazioni random di

forza e momento non causa lo scivolamento dell'oggetto. La qualità appresa approssima implicitamente l'epsilon-quality metric della sezione 28.2, senza che questa debba essere calcolata esplicitamente per ogni configurazione di presa. AnyGrasp estende GraspNet con un Graspness module che predice, per ogni punto della nuvola, la probabilità che quel punto sia un buon punto di contatto (indipendentemente dall'orientamento del gripper), seguita da un modulo di raffinamento dell'orientamento condizionato sul punto di contatto selezionato. Questo schema a due livelli riduce lo spazio di ricerca e permette la generalizzazione a gripper di morfologia diversa senza ri-addestramento, avvicinando la sintesi di grasp al paradigma foundation model.

Capitolo 29 — Architetture Foundation per la Robotica: da RT-2 a GR00T N1

29.1 La traiettoria architetturale: da CLIPort a GR00T N1

La survey sistematica di Kawaharazuka, Oh, Yamada, Posner e Zhu (Tokyo, Oxford, UT Austin, 2025) documenta la traiettoria architettonica dei Vision-Language-Action (VLA) model con una chiarezza che rende evidente la convergenza verso i principi discussi in questa monografia. I VLA sono definiti come sistemi che ricevono osservazioni visive e istruzioni in linguaggio naturale come input obbligatori, producendo comandi di controllo direttamente eseguibili dal robot — in contrasto con i sistemi che usano visione e linguaggio per la pianificazione ad alto livello ma delegano l'esecuzione a primitive preimpostate.

La prima generazione — CNNs (CLIPort, 2021) — combina CLIP per l'estrazione di feature visuo-linguistiche con la Transporter Network per il task di pick-and-place, dimostrando la fattibilità dell'addestramento congiunto visione-linguaggio-azione, ma con capacità di generalizzazione limitata. La seconda generazione — Transformer sequenziali (Gato,

VIMA, RT-1, 2022-2023) — scala il paradigma: Gato usa un singolo transformer decoder per tokenizzare e generare autoregressivamente azioni su 600 task diversi; RT-1 addestra su 130.000 episodi di 700 task con 13 robot Google; RT-2 inizializza il modello con i pesi di PaLM-E o PaLI-X pre-addestrati su dati internet, ereditando la conoscenza enciclopedica del pretraining per il ragionamento zero-shot.

La terza generazione — policy a diffusione e flow matching (Octo, RDT-1B, π0, 2024) — sostituisce la generazione autoregressiva discreta con una testa di diffusione o flow matching, ottenendo azioni continue smoother e più naturali a frequenze di controllo fino a 50 Hz (π0, basato su PaliGemma). La quarta generazione — architetture gerarchiche con latent action e implicit world model (LAPA, π0.5, GR00T N1, 2025) — incorporate esplicitamente rappresentazioni di azioni latenti e predizione di osservazioni future, chiudendo il cerchio con i world model latenti discussi nelle Parti Terza e Quarta.

29.2 World model nei VLA: tre pattern architetturali

La survey di Kawaharazuka et al. identifica tre pattern architetturali per l'incorporazione dei world model nei VLA, corrispondenti a tre modi diversi di usare la predizione futura nella generazione delle azioni. Nel primo pattern — generazione di azione attraverso il world model — il sistema genera sequenze di osservazioni future (immagini o video) e poi usa un modello di dinamica inversa (Inverse Dynamics Model, IDM) per inferire le azioni che producono quella sequenza. UniPi (Du et al. 2023) è il prototipo di questo approccio: usa un diffusion model video per generare la traiettoria visiva ottimale e poi ottimizza le azioni che la realizzano mediante un IDM. Formalmente, dato l'obiettivo in linguaggio naturale g e lo stato iniziale o_0:

$$a_{0:H}\hat{}\,* = IDM(o_{0:H}\hat{}\,*),\ dove\ o_{0:H}\hat{}\,* = arg\,max_{\ o}\, p\left(o_{0:H} \mid o\_0,\ g\right) \quad (20.45)$$

Il secondo pattern — generazione di azione latente attraverso world model (LAPA, GO-1, Moto) — apprende rappresentazioni di azione latente da sequenze di osservazioni senza etichette esplicite, usando il world model come supervisore indiretto. LAPA (Latent Action Pre-training from Videos, Ye et al. 2024) addestra congiuntamente un codificatore di azione latente z_t = Enc(o_t, o_{t+1}) e un world model che predice o_{t+1} a partire da (o_t, z_t), su video umani non annotati. Le azioni latenti apprese sono poi trasferite a policy robotiche mediante fine-tuning. GR00T N1 (NVIDIA, 2025) usa LAPA per il pre-addestramento su video egocentrici umani e poi genera azioni continue mediante un diffusion transformer condizionato sui token del VLM backbone. GR00T N1 incorpora inoltre il COSMOS world model per sintetizzare traiettorie visive in ambienti nuovi a partire da dati simulati, aumentando la diversità del dataset di addestramento.

Il terzo pattern — sensorimotor models con world model impliciti (GR-1, GR-2, GR-3) — genera simultaneamente azioni e predizioni di osservazioni future, incorporando la previsione del futuro come obiettivo ausiliario nell'addestramento della policy. GR-1 (Wu et al. 2023) pre-addestra un MAE-ViT encoder su Ego4D e lo fine-tuna per predire contemporaneamente azioni e frame futuri; la predizione del futuro funge da regolarizzatore che forza la rete a costruire rappresentazioni comprensibili del flusso causale degli eventi, migliorando le performance su compiti long-horizon. Questo terzo pattern è formalmente il più vicino al framework LPA del capitolo 19: la predizione di osservazioni future è esattamente ciò che distingue i world model generativi (con ricostruzione) dalle architetture JEPA (senza ricostruzione).

29.3 Sfide aperte dei VLA e connessioni con i temi della monografia

La survey di Kawaharazuka et al. identifica tre sfide fondamentali dei VLA che si connettono direttamente ai temi trattati in questa monografia. La prima è la scarsità e il costo dei dati visione-linguaggio-azione: i

dataset di dimostrazioni robotiche più grandi (RT-1: 130K episodi, Open-X Embodiment: 1M+ traiettorie da 21 istituzioni, AgiBot World: 1M traiettorie da 100+ robot) restano di diversi ordini di grandezza inferiori ai dataset di pretraining visivo-linguistico su cui si addestrano i VLM backbone. Questa asimmetria motiva l'uso di dati umani (HOI4D, Ego4D) e di generazione sintetica mediante world model come COSMOS, che possono produrre variazioni visive realistiche di traiettorie esistenti.

La seconda sfida è il trasferimento tra embodiment: ogni robot ha spazi di azione e osservazione diversi, e addestrare una singola policy che generalizzi attraverso decine di platform con diverse cinematiche, range articolari e configurazioni di sensori richiede rappresentazioni dell'azione che siano astratte rispetto all'embodiment. Le azioni latenti di LAPA sono un passo in questa direzione, ma la questione fondamentale rimane aperta: esiste una rappresentazione universale dell'intenzione motoria che sia indipendente dall'embodiment specifico? Questa domanda richiama direttamente il problema della composizionalità dell'azione discusso nella sezione 20.2, e suggerisce che le architetture object-centric e le azioni basate su affordance potrebbero fornire la giusta struttura rappresentazionale.

La terza sfida è il costo computazionale di inferenza: i modelli VLA correnti richiedono tipicamente cluster GPU per il training e hardware accelerato per il deployment in tempo reale, limitando la loro deployabilità su piattaforme robotiche con risorse computazionali limitate. Tecniche come Real-Time Chunking (RTC), DeeR-VLA (predizione anticipata a ogni layer del transformer) e BitVLA (quantizzazione a 1.58 bit del vision encoder) indicano che la compressione e l'efficienza dei VLA sono aree di ricerca attive e promettenti. La connessione con il sim-to-real gap è diretta: un VLA deployato on-robot deve non solo essere computazionalmente efficiente ma anche robusto alle discrepanze tra le distribuzioni visive di training

(dataset di dimostrazioni) e quelle di deployment (ambiente reale con illuminazione, texture e viewpoint diversi). Le strategie di augmentazione visiva mediante generative model (GenAug, ROSIE, DreamGen) e le tecniche di representation learning discusse nella sezione 20.3.11 convergono naturalmente verso questa sfida.

La connessione più profonda tra i VLA e il framework teorico di questa monografia è che un VLA completamente realizzato — capace di generalizzare su task, oggetti, ambienti e embodiment — implementa esattamente la visione della robotica cognitiva fondata sulla predizione delineata nella sezione 21.4: un sistema in cui percezione (il VLM backbone che elabora la scena), rappresentazione (le feature latenti visuo-linguistiche), pianificazione (la generazione autoregressiva o diffusiva della sequenza di azioni) e azione (l'esecuzione sul robot) sono mediate da un unico substrato addestrato con obiettivi predittivi su scala massiva. Il percorso da CLIPort a GR00T N1 è, in questo senso, un percorso verso l'implementazione pratica del framework JEPA gerarchico di LeCun — con la differenza non trascurabile che i VLA correnti operano nello spazio dell'immagine e del linguaggio naturale, non in quello delle rappresentazioni astratte nel latente che JEPA propone come paradigma finale. Il passo da VLA che prevedono pixel (UniPi, DreamGen) a VLA che prevedono nello spazio latente (LAPA, GR-1) è il passo dal paradigma generativo al paradigma LPA, e suggerisce una direzione di convergenza tra le due famiglie di architetture che è tra le più promettenti dell'attuale panorama della ricerca.

Capitolo 30 — Validazione, Benchmarking e Divario Simulazione-Realtà

La validazione sperimentale è il momento in cui le architetture teoriche descritte nelle parti precedenti vengono misurate rispetto alla realtà fisica. Questo capitolo unifica i temi già affrontati nelle sezioni 20.3, 13.4 e 13.5 in un trattamento sistematico dei metodi di domain randomization, domain adaptation e sim-to-real transfer, aggiornato con i risultati più recenti e arricchito da una rassegna comparativa dei principali ambienti di simulazione e benchmark di manipolazione.

30.1 Ambienti di Simulazione: Confronto Sistematico

La scelta del simulatore fisico determina sia la fidelità del modello di training sia il costo computazionale della raccolta dati. I quattro simulatori più diffusi in robotica cognitiva sono MuJoCo, IsaacSim, Genesis e PyBullet, ciascuno con caratteristiche diverse in termini di modello di contatto, velocità di simulazione e supporto al training GPU.

MuJoCo (Multi-Joint dynamics with Contact, Todorov et al. 2012) è il simulatore di riferimento per la ricerca in controllo e RL grazie alla sua stabilità numerica e alla velocità su CPU (tipicamente 10^3–10^4 Hz per sistemi a 20–50 gradi di libertà). Il modello di contatto di MuJoCo è basato su compliant contacts con constraint solver lagrangiano: i contatti sono modellati come vincoli anelastici con penetrazione ammessa (β-stabilizzazione), producendo comportamenti di contatto più stabili rispetto ai modelli a impulso rigido. La limitazione principale di MuJoCo è il rendering visivo, non comparabile con i motori grafici in termini di realismo fotorealistico.

NVIDIA IsaacSim (basato su Omniverse, 2022) combina la simulazione fisica GPU-accelerata (PhysX 5 con supporto FEM per deformabili) con il rendering fotorealistico tramite ray tracing in tempo reale. La simulazione parallela su GPU permette di raccogliere dati di training con un throughput di 10^5–10^6 passi al secondo su hardware moderno (A100/H100), riducendo di ordini di grandezza il costo di raccolta dati rispetto alla raccolta fisica. Il framework Isaac Lab (Mittal et al.

2023) fornisce API standardizzate per task di manipolazione, locomozione e navigazione, con supporto a RL, imitation learning e generazione di dati sintetici.

Genesis (Zhou et al. 2024) è un simulatore interamente differenziabile sviluppato per il training di policy con gradient-based optimization. A differenza di MuJoCo e IsaacSim che usano simulatori a fisica discreta (stepping), Genesis implementa una pipeline di simulazione differenziabile end-to-end che permette di backpropagare gradienti dal costo finale attraverso centinaia di passi di simulazione fisica. Questo è essenziale per metodi come l'identificazione differenziabile dei parametri fisici e il training di policy con costi geometrici differenziabili. La limitazione attuale è la velocità di simulazione più bassa rispetto a MuJoCo e la minore maturità del modello di contatto.

30.2 Domain Randomization: Fondamenti e Varianti

La domain randomization (DR, Tobin et al. 2017; Andrychowicz et al. 2020) affronta il divario simulazione-realtà campionando durante il training i parametri del simulatore da distribuzioni ampie, forzando la policy a diventare robusta rispetto alla variabilità dei parametri fisici. Formalmente, la DR massimizza la performance attesa su una distribuzione $P(\xi)$ sui parametri di simulazione ξ:

$$J_{DR}(\pi) = E_{\xi \sim P(\xi)} [E_{\tau \sim \pi,\xi} [\sum_t r(s_t, a_t)]] \quad (30.1)$$

I parametri randomizzati tipicamente includono: proprietà fisiche degli oggetti (massa, attrito, inerzia), proprietà del sensore (delay, rumore, calibrazione), proprietà visive (illuminazione, texture, deformazioni di colore), e parametri del robot (rigidezza degli attuatori, smorzamento, isteresi). La DR uniforme usa $P(\xi) = \prod_i \text{Uniform}(\xi_i^{min}, \xi_i^{max})$, con intervalli impostati manualmente su base empirica. La DR adattativa (Automatic Domain Randomization, ADR, Akkaya et al. 2019) aggiusta automaticamente gli intervalli in base alla performance della policy: se la policy supera una soglia di performance con un dato range, il range viene ampliato; se la performance scende sotto una soglia inferiore, il range viene ristretto.

30.3 Domain Adaptation e Metodi Sim-to-Real

La domain adaptation affronta il divario sim-to-real allineando esplicitamente la distribuzione delle feature estratte dal simulatore con quella estratta dalla realtà. I

metodi più efficaci si dividono in tre categorie: adattamento visivo, identificazione dei parametri fisici, e modelli residuali.

L'adattamento visivo (GraspNet-based DA, CycleGAN, pix2pix) trasforma le osservazioni simulate in osservazioni visivamente realistiche usando reti generative condizionali. Formalmente, si apprende un traduttore T : O_sim → O_real che minimizza una loss avversaria con discriminatore D_real: min_T max_{D_real} E_{o_r}[log D_real(o_r)] + E_{o_s}[log(1 - D_real(T(o_s)))]. Il vantaggio è che si può addestrare T senza coppie simulate-reali allineate (unpaired domain adaptation). La limitazione è che l'adattamento visivo non affronta il divario fisico: una policy addestrata su immagini visivamente realistiche ma con dinamica simulata si trasferisce male se la dinamica reale differisce significativamente.

L'identificazione differenziabile dei parametri fisici (Ajay et al. 2018; Muratore et al. 2022) stima i parametri ξ del simulatore minimizzando la discrepanza tra traiettorie simulate e traiettorie reali:

$$\xi^* = \arg\min_{\xi} D_{KL}(p(\tau \mid \xi) \,\|\, p_{real}(\tau)) \quad (30.2)$$

I modelli residuali (Hanna e Stone 2017; Johannink et al. 2019) apprendono la differenza tra la dinamica simulata e quella reale come addendum Δf al modello nominale: f_real(s,a) = f_sim(s,a) + Δf(s,a), dove Δf è una rete neurale addestrata su pochi dati reali. Questo approccio è efficiente in termini di dati reali richiesti ed è compatibile con i world model della Parte III: il modello residuale può essere interpretato come un correttore del world model latente addestrato in simulazione.

30.4 Benchmark di Manipolazione: Tassonomia e Confronto

I benchmark di manipolazione forniscono il substrato sperimentale su cui vengono valutati i sistemi descritti nella monografia. La tassonomia più utile distingue i benchmark in base al tipo di task (singolo oggetto vs. multi-oggetto vs. articolato), al tipo di supervisione (RL, imitation learning, zero-shot), e alla disponibilità di setup fisico.

robosuite (Zhu et al. 2020) è un framework Python basato su MuJoCo che fornisce una suite standardizzata di task di manipolazione con API uniformi per RL e imitation learning. I task includono Lift (presa di un cubo), PickPlace (presa e posizionamento in una cesta), NutAssembly (avvitamento di un dado su un perno

con cinematica vincolata), TwoArmHandover (passaggio di un oggetto tra due bracci coordinati), e Wipe (pulizia di una superficie con il robot mobile). La disponibilità di dimostrazioni di riferimento, la standardizzazione delle metriche (success rate, sample efficiency) e la profondità di osservazione (image, proprio, object state) rendono robosuite il benchmark più diffuso per i confronti in manipolazione.

FurnitureBench (Heo et al. 2023) è un benchmark di lunga distanza ispirato all'assemblaggio di mobili IKEA, con task che richiedono decine di passi sequenziali di manipolazione (avvitamento, inserimento, allineamento). La difficoltà principale di FurnitureBench è la gestione degli errori di lunga traiettoria: un errore di allineamento nelle fasi iniziali si propaga e amplifica nelle fasi successive, rendendo il task un banco di prova per la pianificazione gerarchica e il recovery da errori. RLBench (James et al. 2020) è un benchmark multi-task con oltre 100 task di manipolazione di varia difficoltà, progettato per valutare la generalizzazione multi-task e il few-shot learning con descrizioni linguistiche.

LIBERO (Liu et al. 2023) è specificamente progettato per valutare il trasferimento di conoscenza nell'imitation learning: la suite include suite di task con trasferimento spaziale (stessa task, ambiente diverso), semantico (task diverso, oggetti simili), strutturale (layout diverso) e full. Questa tassonomia consente di misurare esplicitamente la generalizzazione dei modelli rispetto alle diverse forme di distributional shift, connettendosi direttamente con la teoria dei world model e con il framework JEPA: un sistema con una buona rappresentazione latente dovrebbe trasferirsi meglio su LIBERO-Spatial rispetto a un sistema senza world model, poiché la rappresentazione latente cattura la struttura geometrica dell'ambiente indipendentemente dalla configurazione specifica.

30.5 Digital Twin e Validazione in Loop Chiuso

Il digital twin (Grieves 2014) è un modello computazionale aggiornato in tempo reale che replica la stato di un sistema fisico con sufficiente fidelità da permettere la previsione del suo comportamento futuro. In robotica, il digital twin di un robot industriale include il modello cinematico e dinamico calibrato, il modello 3D dell'ambiente di lavoro aggiornato da sensori (scanner 3D, fotocamere RGB-D), e il modello dei processi (velocità di usura degli utensili, deriva delle giunzioni). La validazione in loop chiuso su digital twin precede il deployment fisico: le policy addestrate in simulazione vengono eseguite sul digital twin (che differisce dal

simulatore di training per la presenza di rumore sensoriale reale e di parametri fisici identificati in situ) e testate contro scenari di fallimento progettati, prima del trasferimento sul robot fisico.

La connessione tra digital twin e world model latenti chiude il ciclo dell'intera monografia: il world model latente φ addestrato su dati di simulazione e su dati reali è il componente che avvicina il simulatore a un digital twin, incorporando la storia delle osservazioni reali come condizionamento implicito della predizione futura. Il passaggio dai simulatori fisici basati su equazioni differenziali ordinarie (MuJoCo, Genesis) ai world model neurali basati su dati (DreamerV3, LAPA) rappresenta la traiettoria tecnologica verso sistemi in grado di costruire autonomamente il proprio digital twin dall'esperienza sensoriale, realizzando la visione della robotica cognitiva fondata sulla predizione che costituisce il filo conduttore di questa monografia.

Capitolo 31 — Sintesi Finale e Prospettive

31.1 Il Filo Conduttore: dalla Geometria del Corpo alla Geometria della Rappresentazione

Le cinque parti di questa monografia hanno tracciato un percorso che, a una lettura retrospettiva, rivela una coerenza geometrica profonda. La Parte Prima ha fondato la cinematica e la dinamica del robot sulle strutture di gruppo di Lie $SO(3)$ e $SE(3)$, mettendo in luce come lo spazio delle configurazioni sia intrinsecamente una varietà curva non euclidea e come la correttezza degli algoritmi dipenda dal rispetto di questa curvatura. La Parte Seconda ha sviluppato la teoria degli approssimatori neurali, del reinforcement learning e dei modelli generativi, mostrando come le reti neurali profonde approssimino funzioni in spazi di Barron con tassi indipendenti dalla dimensione e come i modelli variazionali costruiscano rappresentazioni latenti strutturate. Le Parti Terza e Quarta hanno analizzato rispettivamente i world model latenti (nella famiglia RSSM-Dreamer) e l'architettura JEPA di LeCun, entrambi fondati sul

paradigma della predizione nel latente come alternativa alla generazione esplicita.

Il capitolo 19 ha reso esplicita l'unità strutturale di queste architetture attraverso il framework LPA (Latent Predictive Architecture): JEPA e RSSM sono istanze della stessa classe astratta di architetture predittive nel latente, che differiscono essenzialmente per la presenza o assenza del termine di ricostruzione e per il condizionamento sull'azione. Questo riconoscimento ha aperto la strada all'architettura AC-JEPA, che combina la struttura causale condizionata dall'azione dei world model con l'efficienza delle rappresentazioni non generative di JEPA.

Il Capitolo 26, considerevolmente più lungo e articolato nell'edizione presente rispetto alla prima versione, ha affrontato le tre sfide aperte che si frappongono tra le architetture analizzate e la loro applicazione robotica: la causalità nelle rappresentazioni latenti, la composizionalità strutturale e il divario simulazione-realtà. Per ciascuna sfida, la trattazione ha percorso il tragitto completo dalla formalizzazione matematica alle soluzioni sperimentalmente validate nella letteratura più recente (2025-2026), includendo il framework bi-livello per la Sim2Real ottimalità, i digital twin dinamici del paradigma Real-is-Sim, l'identificazione differenziabile HALO, la regolarizzazione guidata dalla fisica, il trasferimento per imitazione dell'MPC e la manipolazione destra force-sensitive.

31.2 Simmetrie Strutturali tra le Parti

Una lettura trasversale della monografia rivela simmetrie strutturali tra concetti apparentemente distanti. Il ruolo dello Jacobiano cinematico J(q) nella Parte Prima — come mappa lineare che trasforma velocità articolari in velocità operazionali — trova il suo analogo perfetto nello Jacobiano della rete neurale $J_{NN} = \partial f/\partial \theta$ nelle analisi di Neural Tangent Kernel della Parte Seconda. In entrambi i casi, le singolarità della mappa (rango deficiente di J e J_{NN}) segnalano una perdita di informazione: nel primo

caso la perdita di mobilità del robot in alcune direzioni dello spazio dei compiti, nel secondo la perdita di capacità espressiva della rete in alcune direzioni dello spazio dei parametri.

Analogamente, la funzione di Lyapunov V(e, Ẽ) che garantisce la stabilità del controllo PD nella Parte Prima è strutturalmente identica all'ELBO variazionale della Parte Seconda e all'obiettivo di addestramento dell'RSSM nella Parte Terza: tutti e tre sono funzionali che bilanciano un termine di adattamento ai dati (l'errore di inseguimento, il termine di ricostruzione, la log-verosimiglianza) con un termine di regolarizzazione (la dissipazione del controllo derivativo, la divergenza KL dal prior, la divergenza KL tra posterior e prior dinamico). La struttura porta-hamiltoniana del sistema meccanico — passività, conservazione dell'energia nella dinamica libera, dissipazione nel controllo — riappare nelle proprietà di regolarizzazione variazionale dei world model.

Nel Capitolo 26, il gap di prestazione G_perf($\mathcal{M}$_s, $\mathcal{M}$_r, π) tra simulazione e realtà — definito come la differenza nei ritorni attesi — rivela un'ulteriore simmetria con il bound di generalizzazione della Parte Seconda: in entrambi i casi, si dimostra che il gap di distribuzione tra ambiente di addestramento e ambiente di test (training distribution shift) non può essere limitato solo mediante la minimizzazione della perdita in-distribuzione, ma richiede strategie esplicite di robustezza o adattamento. La domain randomization è l'analogo robotico della data augmentation in computer vision: aggiunge variabilità artificiale per estendere il supporto della distribuzione di addestramento. Il framework bi-livello di Anand et al. è l'analogo robotico dell'ottimizzazione consapevole del dominio (domain-aware optimization): invece di minimizzare la perdita su tutti i possibili ambienti, si ottimizza direttamente per le prestazioni nel dominio target.

31.3 Le Frontiere Aperte: una Mappa dei Problemi Irrisolti

Nonostante i progressi notevoli documentati in questa monografia, le frontiere della ricerca rimangono ampie e i problemi fondamentali irrisolti sono numerosi. Si elencano di seguito le sfide che a giudizio dell'autore rappresentano i nodi più profondi da sciogliere nel prossimo decennio.

La causalità nelle rappresentazioni latenti rimane la sfida più profonda: costruire encoder che producano rappresentazioni causalmente disentangled senza supervisione sul grafo causale è un problema di identificabilità che ha ricevuto soluzioni parziali (Khemakhem et al. 2020, Von Kügelgen et al. 2021) ma non ha ancora una soluzione generale per l'ambito visivo non strutturato. La distanza tra i teoremi di identificabilità e la praticabilità computazionale rimane enorme, e la definizione operativa di 'variabile causale' in un sistema visivo continuo è ancora aperta.

La composizionalità scalabile — la capacità di riconoscere, ragionare e pianificare su scene visive mai viste come combinazioni di concetti noti — è posseduta solo parzialmente dai sistemi neurali attuali. I metodi oggetto-centrici hanno compiuto progressi significativi, ma la scalabilità a scene del mondo reale con centinaia di oggetti, occlusioni dense e variabilità visiva estrema rimane un problema aperto. La connessione tra composizionalità visiva e composizionalità del ragionamento linguistico, esplorata dai modelli multimodali come GPT-4V e Gemini, suggerisce che l'integrazione di rappresentazioni visive e linguistiche possa fornire un percorso verso la composizionalità generalizzata.

Il divario simulazione-realtà, pur avendo ricevuto mitigazioni efficaci attraverso le strategie analizzate nel Capitolo 26, rimane un ostacolo strutturale per la robotica di precisione. In compiti che richiedono tolleranze sub-millimetriche, come l'assemblaggio di componenti elettronici o la chirurgia robotica, la dinamica di contatto che i simulatori non catturano è determinante per il successo. Le soluzioni più promettenti combinano simulatori fisici ad alta fidelità, modelli residuali apprendibili,

identificazione differenziabile, regolarizzazione guidata dalla fisica e digital twin dinamici — ma la scalabilità di queste combinazioni a sistemi con molti gradi di libertà, molti oggetti e dinamica di contatto complessa rimane largamente inesplorata.

Il problema dell'adattamento on-line — l'aggiornamento del world model durante il deployment, senza accesso a scenari sicuri di raccolta dati e senza degradare le skill già acquisite (il problema del catastrophic forgetting nel continual learning) — è uno dei problemi più praticamente rilevanti e meno risolti. I meccanismi di adattamento rapido come RMA funzionano su variazioni moderate dei parametri fisici, ma l'adattamento a cambi strutturali dell'ambiente (un nuovo tipo di superficie, un oggetto di massa molto diversa da quelle incontrate in training) rimane una sfida aperta.

31.4 Verso una Robotica Cognitiva Fondata sulla Predizione

La visione a lungo termine che emerge dalle architetture analizzate in questa monografia è quella di un sistema robotico in cui la percezione, la rappresentazione, la pianificazione e l'azione sono mediate da un unico substrato di predizione nel latente — una gerarchia di modelli predittivi che operano a scale temporali e spaziali diverse, ciascuno apprendendo ad anticipare l'evoluzione delle rappresentazioni al livello sottostante. Questa visione, articolata da LeCun nel framework JEPA gerarchico e indipendentemente perseguita dai ricercatori di Dreamer e dei world model gerarchici, condivide radici profonde con la teoria della predictive coding di Rao e Ballard e con il free energy principle di Friston.

In questa architettura unificata, il confine tra percezione e pianificazione si dissolve: percepire significa costruire un modello predittivo della scena, e pianificare significa simulare traiettorie nel latente del modello predittivo. Il corpo del robot — con la sua geometria di Lie, le sue equazioni del moto lagrangiane, i suoi attuatori e sensori — è il substrato fisico che

implementa le azioni selezionate dalla pianificazione latente e fornisce le osservazioni che alimentano l'aggiornamento del modello. La comprensione matematica di questo substrato, sviluppata nelle prime due parti di questa monografia, è la condizione necessaria — anche se non sufficiente — per costruire sistemi che non solo apprendano a muoversi nel mondo fisico, ma lo comprendano nel senso più profondo: siano in grado di anticiparne l'evoluzione, di ragionare sulle conseguenze delle proprie azioni, di adattarsi a perturbazioni impreviste attraverso un ciclo continuo di predizione, percezione e correzione.

Il sim-to-real gap — analizzato in profondità nel Capitolo 26 con un panorama che va dalla formalizzazione POMDP ai framework bi-livello più recenti — non è un problema tecnico residuale destinato a sparire con simulatori più fedeli. È la manifestazione di un problema più profondo: la difficoltà di apprendere modelli del mondo che siano non solo predittivamente accurati in distribuzione, ma causalmente corretti, compositivamente organizzati e robusti alla distribuzione shift. Finché le rappresentazioni latenti saranno puramente correlazionali — senza struttura causale, senza organizzazione oggetto-centrica, senza distinzione tra varianza in-distribuzione e varianza out-of-distribution — il gap rimarrà una frontiera strutturale, non un semplice artefatto ingegneristico.

La distanza che separa i sistemi attuali da questa visione è ancora considerevole. I world model attuali operano su orizzonti temporali di decine di secondi, su compiti con obiettivi definiti e ambienti semi-strutturati. La scalabilità a comportamenti multiminuto, a obiettivi gerarchici e a ambienti completamente aperti — il vero mondo fisico — richiede progressi fondamentali non solo nell'architettura dei modelli, ma nella comprensione teorica di cosa significhi per una macchina apprendere un modello del mondo che sia causalmente corretto, compositivamente ricco e trasferibile tra domini diversi. Il framework bi-livello di Anand et al., i digital twin di Abou-Chakra et al.,

l'identificazione differenziabile di Wang et al. e la regolarizzazione guidata dalla fisica di Kawachi sono contributi che, presi insieme, delineano una direzione coerente: non costruire il simulatore perfetto, ma costruire il sistema di apprendimento che impara a bridgare il gap qualunque sia la sua origine.

Il percorso tracciato in questa monografia — dai gruppi di Lie della cinematica alle architetture predittive nel latente, passando per le teorie dell'approssimazione neurale e del reinforcement learning — è un percorso che non termina con le conclusioni di questa quinta parte. Termina, come ogni percorso intellettuale rigoroso, di fronte a una frontiera — e la frontiera, in questo caso, è tanto vasta quanto promettente.

31.5 World Model per l'Interazione Umano-Oggetto: da HOI4D a InterDreamer

31.5.1 Il problema dell'interazione umano-oggetto e la sua formalizzazione

L'interazione umano-oggetto (Human-Object Interaction, HOI) rappresenta il livello semantico più ricco e strutturalmente complesso dell'azione robotica: l'agente non manipola un oggetto isolato in condizioni controllate, ma si inserisce in un flusso di interazioni multi-parte, dove la mano umana — o l'end-effector del robot — modifica la posa e lo stato di un oggetto articolato o deformabile in modo contestualmente dipendente dalla storia dell'interazione e dall'intenzione dell'agente. Comprendere, generare e imitare interazioni di questo tipo richiede modelli che uniscano la semantica ad alto livello — cosa si sta facendo e perché — con la dinamica fisica a basso livello — come la mano esercita forze sull'oggetto e come l'oggetto risponde nel tempo.

Il dataset HOI4D, introdotto da Liu et al. (Tsinghua University e Peking University, 2024), affronta la carenza di dati annotati per questa categoria di interazioni con una raccolta di scala senza precedenti: 2.4 milioni di frame RGB-D egocentrici distribuiti su 4.000 sequenze, acquisite da 9 partecipanti che interagiscono con 800 istanze di oggetti appartenenti a 16 categorie — comprendendo sia oggetti rigidi sia articolati — in 610 ambienti interni diversi, senza markers artificiali. Le annotazioni frame-wise coprono: segmentazione panotica, segmentazione del moto, posa 3D della mano, posa dell'oggetto a livello categoriale (non solo d'istanza), segmentazione dell'azione, e mesh ricostruite degli oggetti con le point cloud della scena.

La distinzione tra HOI a livello d'istanza e HOI a livello categoriale è cruciale per la generalizzazione: i dataset precedenti (H2O, DexYCB, HO-3D) operavano con un pool ristretto di oggetti di cui si conosce a priori il modello CAD e le dimensioni esatte, rendendo impossibile la generalizzazione a oggetti nuovi della stessa categoria. HOI4D, operando a livello categoriale, consente di apprendere rappresentazioni che generalizzano alle variazioni intra-categoria di forma, dimensione e articolazione, avvicinando le condizioni del dataset a quelle del deployment reale. Il problema è formalizzato come segue: dato un oggetto appartenente alla categoria c con posa $\pi\_t = (R\_t, t\_t) \in SE(3)$ al tempo t, il task di category-level object pose tracking stima la sequenza di pose $\{\pi\_1, ..., \pi\_T\}$ a partire dal flusso RGB-D, senza conoscere il modello CAD dell'istanza specifica:

$$\pi^{*}_{1:T} = argmax_{\pi\_\{1:T\}} p(\pi\_\{1:T\} \mid RGB - D\_\{1:T\}, c) \quad (20.38)$$

I benchmark istituiti da HOI4D sono tre: (i) semantic segmentation di sequenze di point cloud 4D dinamiche, che richiede di classificare ogni punto in ogni frame tra 16 categorie in presenza di occlusioni dense ed ego-motion rapido; (ii) category-level object pose tracking per interazioni

hand-object, dove la mano occlude parzialmente l'oggetto durante tutta l'interazione; (iii) egocentric hand action recognition su target di interazione diversi, che richiede di riconoscere azioni fine-grained (reach, grasp, move, place, open, close) su oggetti categorialmente diversi. La difficoltà intrinseca dei benchmark riflette la sfida aperta della 4D scene understanding in presenza di interazioni reali: i metodi state-of-the-art mostrano performance significativamente degradate rispetto a dataset di navigazione outdoor, dove le condizioni sono meno caotiche e le occlusioni meno dense.

31.5.2 Il world model per la dinamica delle interazioni: InterDreamer

Liu et al. avevano dimostrato con HOI4D che i dati per le interazioni umano-oggetto sono scarsi e costosi da annotare. Xu, Wang e colleghi (UIUC, NeurIPS 2024, aggiornato 2026) affrontano il problema della generazione di interazioni HOI testo-condizionate senza dati di addestramento appaiati testo-interazione, attraverso il framework InterDreamer. L'intuizione centrale è la separabilità tra semantica e dinamica dell'interazione: la semantica ad alto livello — cosa si sta facendo con quale oggetto — può essere estratta da modelli pre-addestrati su dati di moto umano e da LLM con conoscenza enciclopedica; la dinamica a basso livello — come l'oggetto si muove sotto l'influenza della mano umana — può essere appresa da dati mocap di HOI senza annotazioni testuali.

Formalmente, InterDreamer modella il problema come un MDP gerarchico: il livello alto (pianificazione) usa l'LLM e il modello testo-moto per generare la sequenza di pose umane H = {h_1, ..., h_M} e la posa iniziale dell'oggetto o_1, dove h_i ∈ ℝ^{85} è il vettore dei parametri SMPL; il livello basso (controllo) usa il world model per propagare la posa dell'oggetto nel tempo dati i contatti con la mano:

$$o_{t+1} = f_\theta(o_t, a_t, s_t) \quad (20.39)$$

dove o_t ∈ SE(3) è la posa dell'oggetto al tempo t, a_t è l'azione del world model (i vertici del corpo umano nella zona di contatto), e s_t è lo stato latente che codifica le dinamiche fisiche dell'interazione. La chiave dell'approccio è l'astrazione dell'azione: invece di usare l'intera mesh del corpo umano (che dipende fortemente dall'identità del soggetto e dalla forma dell'oggetto), InterDreamer rappresenta l'azione mediante soli i vertici della mesh SMPL che si trovano nella zona di contatto rilevata. Questa astrazione rende il world model indipendente dalla forma globale dell'oggetto e dalla cinematica complessiva del corpo, consentendo la generalizzazione zero-shot a dataset e categorie di oggetti non visti durante l'addestramento (BEHAVE → OMOMO, CHAIRS).

L'obiettivo di addestramento del world model è la predizione della trasformazione rigida ΔT_t ∈ SE(3) che porta la posa dell'oggetto da o_t a o_{t+1}, condizionata sui vertici di contatto v_t e sullo stato latente s_t, addestrato mediante regressione sulla ground truth:

$$\theta^{*} = argmin_{\theta} \frac{1}{T} \sum_{t=1}^{T} \| \Delta T_t - f_{\theta}(v_t, s_t) \|^{2} \quad (20.40)$$

Il risultato chiave di InterDreamer — la possibilità di generare interazioni umano-oggetto tridimensionali realistiche e allineate con il testo senza nessun dato appaiato testo-interazione durante l'addestramento del world model — illustra un principio più generale di grande rilievo per la robotica: la conoscenza fisica può essere disaccoppiata dalla conoscenza semantica e appresa da fonti diverse, per essere poi ricombinata a runtime. Questo principio corrisponde alla visione gerarchica di JEPA descritta nel capitolo 19, dove livelli diversi della gerarchia apprendono aspetti diversi del mondo a diverse scale temporali e semantiche.

31.5.3 Manipolazione deformabile: la frontiera aperta del contatto complesso

La manipolazione di oggetti deformabili — tessuti, corde, pasta, materiali biologici — rappresenta la frontiera più aperta della robotica di manipolazione e la fonte di divario sim-to-real strutturalmente più profonda. I simulatori di corpi rigidi, per quanto accurati, non possono catturare la meccanica della deformazione elastica, plastica e viscosa senza ricorrere a metodi di elementi finiti (FEM) o di particelle (MPM, SPH), che sono troppo lenti per il ciclo di addestramento RL e producono comunque approssimazioni imperfette dei materiali reali.

La survey di Kawaharazuka et al. (2025) sui VLA identifica la manipolazione di oggetti deformabili come una delle sfide benchmark più usate nei sistemi VLA avanzati: ManiSkill (SAPIEN) include task di piegatura di tessuti e manipolazione di oggetti articolati; RoboTwin include 50 task di manipolazione duale con 731 oggetti che comprendono categorie deformabili. Il trattamento formalizzato di questa classe di problemi richiede di modellare lo stato dell'oggetto come un campo di deformazione continuo φ : ℝ^3 → ℝ^3 piuttosto che come una posa rigida $\pi \in$ SE(3). Un world model per oggetti deformabili deve predire come il campo di deformazione evolve sotto l'applicazione di forze distribuite f(x, t) da parte dell'end-effector:

$$\varphi_{t+\delta t}(x) \;=\; \varphi_t(x) \;+\; \delta t \cdot G_\theta\big(\varphi_t,\, f(x,\, t)\big) \quad (20.41)$$

dove G_θ è il predittore dinamico (tipicamente una rete a grafo o una rete operante su point cloud) che modella l'influenza delle forze applicate sul campo di deformazione. La difficoltà fondamentale di questo approccio è che lo spazio degli stati φ è infinito-dimensionale (un campo continuo), e la sua discretizzazione in una point cloud o in una mesh introduce errori che si accumulano su orizzonti temporali lunghi. Le direzioni di ricerca più promettenti combinano rappresentazioni neurali implicite dello stato deformabile (Neural Implicit Fields) con world model basati su grafi che rispettino l'invarianza alle simmetrie fisiche, aprendo la strada a una

prossima generazione di sistemi robotici in grado di manipolare la materia morbida con la stessa competenza con cui i sistemi attuali manipolano oggetti rigidi.

31.6 Percezione 3D Continua con Stato Latente Persistente: CUT3R

31.6.1 Il modello CUT3R come world model percettivo

Wang, Zhang, Holynski, Efros e Kanazawa (UC Berkeley e Google DeepMind, 2025) hanno proposto CUT3R (Continuous Updating Transformer for 3D Reconstruction), un framework unificato per la percezione 3D online che mantiene e aggiorna uno stato latente persistente a ogni nuova osservazione visiva. La struttura del sistema è formalmente analoga a quella di un world model percettivo: lo stato latente S_t codifica la comprensione accumulata della scena 3D fino al tempo t, e viene aggiornato con ogni nuovo frame I_t attraverso un meccanismo di interazione stato-osservazione.

L'encoder ViT trasforma il frame corrente I_t in token F_t = Encoder(I_t) $\in \mathbb{R}$^{N_p × d}, dove N_p è il numero di patch e d è la dimensione delle feature. I token dell'immagine interagiscono con lo stato S_t $\in \mathbb{R}$^{N_s × d} in due direzioni: aggiornano lo stato con le nuove informazioni osservate, e recuperano dallo stato le informazioni necessarie per predire la geometria 3D del frame corrente. Il meccanismo di interazione bidirezionale è implementato mediante cross-attention:

$$S_{t+1} = f_{up}(S_t, F_t) \quad (20.42)$$

$$P_t = f_{read}(S_{t+1}, F_t) \quad (20.43)$$

dove P_t $\in \mathbb{R}$^{H × W × 3} è la pointmap predetta per il frame t — una mappa che associa a ogni pixel le coordinate 3D del punto di superficie corrispondente in un sistema di coordinate globale. Le pointmap possono

essere accumulate nel tempo per produrre una ricostruzione densa della scena che si raffina con ogni nuova osservazione, senza richiedere post-processing offline né allineamento globale esplicito. Lo stato S_{t+1} può anche essere interrogato con una vista virtuale (una raymap parametrica) per predire la geometria di regioni non ancora osservate, realizzando la capacità di completamento della scena analoga all'inferenza su regioni mascherate di JEPA.

31.6.2 Connessioni formali con i world model robotici

La struttura di CUT3R rivela connessioni profonde con i world model robotici discussi nei capitoli precedenti. Il meccanismo di aggiornamento ricorrente dello stato S_t è formalmente identico al modello di transizione dell'RSSM di Dreamer: in entrambi i casi, uno stato latente viene aggiornato a ogni passo temporale mediante l'osservazione corrente, e lo stato aggiornato costituisce la rappresentazione compressiva dell'intera storia delle osservazioni passate. La differenza principale è il dominio: mentre l'RSSM modella dinamiche di controllo (stati del robot, effetti delle azioni), CUT3R modella dinamiche percettive (geometria 3D della scena, posa della telecamera).

La capacità di CUT3R di interrogare regioni non osservate mediante viste virtuali corrisponde al meccanismo di predizione nel latente di JEPA: in entrambi i casi il modello impara a completare le informazioni mancanti a partire dal contesto disponibile, senza generare pixel ma predendo rappresentazioni strutturate (pointmap in CUT3R, feature latenti in JEPA). Questa corrispondenza suggerisce che CUT3R è un'istanza del framework LPA (Latent Predictive Architecture) introdotto nel capitolo 19, specializzata per la geometria 3D piuttosto che per il controllo.

Per la robotica, il contributo pratico di CUT3R è la disponibilità di uno stato 3D della scena sempre aggiornato e metricamente coerente, che può servire come input al world model robotico in sostituzione delle costose

pipeline di ricostruzione offline basate su SfM o SLAM. In questo senso, CUT3R realizza il componente percettivo di un sistema robotico cognitivo completo: la percezione produce uno stato latente persistente della scena 3D, e il world model robotico opera su questo stato per pianificare le azioni. La separazione tra percezione 3D (CUT3R) e world model di controllo (RSSM/AC-JEPA) corrisponde alla separazione modulare che la teoria predictive coding identifica tra i livelli inferiori della gerarchia corticale (che elaborano la geometria sensoriale) e i livelli superiori (che pianificano le azioni).

31.6.3 Scene dinamiche e il problema del temporal world model

CUT3R estende il framework anche alle scene dinamiche, dove oggetti in movimento devono essere ricostruiti insieme alla geometria statica della scena. In questo caso, lo stato S_t deve codificare sia la struttura geometrica della scena sia il moto degli oggetti mobili, riportando il problema nella zona di confine tra percezione 3D e world model dinamico. Formalmente, la pointmap P_t per scene dinamiche deve separare le componenti statiche e dinamiche:

$$P_t = P_t^{static} + P_t^{dyn} \quad (20.44)$$

La decomposizione statico/dinamico è un problema aperto in CUT3R come in tutti i sistemi di percezione 3D continua: i metodi attuali usano il flusso ottico o la coerenza temporale delle predizioni come segnale di segmentazione, ma la robustezza su scene con molti oggetti in movimento e occlusioni dense rimane limitata. L'integrazione di HOI4D come dataset di pre-addestramento per CUT3R — dove le annotazioni di moto segmentation forniscono etichette per la componente dinamica della scena — è una direzione naturale che potrebbe chiudere il circolo tra dataset di interazione, percezione 3D e world model, aprendo la strada a sistemi robotici che capiscono simultaneamente cosa si trova nella scena

(geometria 3D), come si muove (dinamica 3D) e come interagire con esso (world model di controllo).

31.7 La Parte Sesta nel Quadro Complessivo: Chiusura del Ciclo

La Parte Sesta, aggiunta in questa edizione per rispondere a una lacuna strutturale segnalata nell'analisi critica, completa il cerchio tra i fondamenti formali della Parte Prima e le architetture predittive delle Parti III–V. Il Capitolo 27 ha ricondotto la pianificazione del moto alla topologia dello spazio delle configurazioni C_free, mostrando come RRT*, PRM e CHOMP siano algoritmi che operano su varietà differenziabili la cui struttura di Lie (SO(3), SE(3)) è esattamente quella fondamentale della Parte Prima. Il planning differenziabile nel latente (sezione 27.5) ha poi istituito il collegamento diretto con i world model della Parte III: ottimizzare una traiettoria nel latente di un modello RSSM è un CHOMP su C_latente, con la differenza che la SDF degli ostacoli è sostituita dal costo implicito appreso dal world model.

Il Capitolo 28 ha fondato il grasping sulla meccanica del contatto (cono di attrito di Coulomb, force closure, epsilon-quality metric) e il controllo di impedenza sulla struttura porto-hamiltoniana della dinamica del robot — la stessa struttura già incontrata nel Capitolo 3 e nel Capitolo 6. La passività garantita del controllo di impedenza è una proprietà di Lyapunov: il funzionale $H(q,p) + V_d(x_e)$ è un candidato di Lyapunov per il sistema in anello chiuso, e la scelta $D_d \geq 0$ ne garantisce la decrescenza per qualunque ambiente passivo con cui il robot interagisce. Questa è la stessa logica con cui il Capitolo 4 ha costruito la stabilità del controllo PD, e con cui il Capitolo 6 ha garantito la convergenza del controllo geometrico su SO(3).

Il Capitolo 30, che ha integrato le sezioni orfane sull'architettura VLA promuovendole a capitolo autonomo, ha completato la tassonomia delle

quattro generazioni di modelli vision-language-action (CNN, Transformer autoregressivo, policy a diffusione, architetture gerarchiche con latent action) e ha mostrato come i VLA di quarta generazione — GR00T N1, LAPA, GR-1 — implementino concretamente il framework LPA del Capitolo 19: il backbone visuo-linguistico svolge il ruolo del contesto, il modello di predizione latente svolge il ruolo del world model, e la politica di decodifica in azioni svolge il ruolo del controller. Il Capitolo 31, integrando §20.3 con il confronto sistematico dei simulatori e dei benchmark, ha fornito il substrato empirico che mancava all'intera trattazione, consentendo di misurare concretamente i progressi e le lacune dei sistemi analizzati.

La simmetria strutturale che la sezione 31.2 ha individuato tra il formalismo della Parte Prima e quello delle Parti III–IV si approfondisce con l'aggiunta della Parte Sesta. Lo Jacobiano cinematico J(q) che compare nel controllo di impedenza (Capitolo 29) è lo stesso che compare nella formula del prodotto di esponenziali (Capitolo 5) e nell'analisi di manipolabilità. Il cono di attrito di Coulomb (sezione 28.1) è un vincolo convesso sullo spazio delle forze, strutturalmente analogo ai vincoli di positivity che compaiono nei CBF (Capitolo 7) per la sicurezza. I benchmark di manipolazione del Capitolo 30 (robosuite, FurnitureBench, LIBERO) sono gli ambienti in cui tutte le architetture analizzate — RSSM, JEPA, VLA — devono dimostrare la propria utilità pratica. La monografia è ora un percorso completo: dai fondamenti geometrici dell'azione al substrato empirico della validazione, passando per le architetture predittive che connettono percezione, rappresentazione e controllo.

PARTE VI

Sintesi e Frontiere

Sull'unità concettuale dei modelli predittivi nel latente, le sfide aperte della causalità e della composizionalità, e le prospettive per una robotica cognitiva fondata su rappresentazioni astratte dell'azione.

Capitolo 25 — Integrazione: World Models, JEPA e Robotica

25.1 Convergenza Epistemica: il Predittivo nel Latente come Paradigma Unificante

Le tre architetture al centro di questa monografia — i world model nella formulazione RSSM-Dreamer, l'architettura JEPA di LeCun, e i sistemi di controllo robotico basati sull'apprendimento — sono state sviluppate da comunità di ricerca in parte distinte, con motivazioni parzialmente diverse e con un lessico tecnico che riflette tali differenze. Eppure, una lettura attenta del loro formalismo matematico rivela un'unità di struttura che non è superficiale: tutte e tre convergono verso il paradigma della predizione nel latente come meccanismo fondamentale di apprendimento e pianificazione.

La tesi che questo capitolo intende argomentare in modo formale è la seguente: JEPA e i world model del tipo RSSM sono due istanze di una medesima classe astratta di architetture predittive nel latente, che differiscono essenzialmente per due aspetti — la presenza o assenza di un termine di ricostruzione nell'obiettivo di addestramento, e la presenza o assenza di un modello esplicito delle azioni come variabile condizionante. Comprendere questa relazione non è meramente un esercizio di tassonomia: apre la strada a una genuina integrazione delle due famiglie di architetture in sistemi robotici che combinano la ricchezza rappresentazionale dei world model con l'efficienza computazionale e la robustezza delle predizioni nel latente di JEPA.

Genesi del paradigma: dall'autocodifica variazionale alla predizione nel latente

La strada che conduce ai world model moderni e a JEPA attraversa una sequenza di rivoluzioni architetturali successive, ciascuna delle quali ha introdotto un pezzo del puzzle. Il primo pezzo è il Variational Autoencoder (VAE) di Kingma e Welling (2013): la scoperta che una rete neurale può apprendere rappresentazioni latenti strutturate e generative minimizzando l'ELBO, il lower bound variazionale sulla log-verosimiglianza dei dati. Il VAE ha formalizzato l'obiettivo di apprendimento di rappresentazioni attraverso il framework dell'inferenza variazionale, introducendo il reparametrization trick che permette di propagare i gradienti attraverso il campionamento stocastico.

Il secondo pezzo è il World Model di Ha e Schmidhuber (2018), che ha esteso il VAE al regime temporale delle sequenze di osservazioni raccolte da un agente in movimento: le osservazioni consecutive di un robot o di un agente in un videogioco sono fortemente correlate, e un modello del mondo deve catturare questa struttura temporale per poter simulare traiettorie future utili alla pianificazione. Ha e Schmidhuber combinano un VAE per la compressione delle osservazioni in rappresentazioni latenti compatte con un modello ricorrente MDN-RNN (Mixture Density Network - Recurrent Neural Network) per le dinamiche nello spazio latente. Il piano nell'immaginazione — la generazione di traiettorie immaginarie e la selezione di azioni basata su esse — viene realizzato interamente nello spazio latente del VAE, senza mai tornare all'osservazione.

Il terzo pezzo è l'architettura RSSM di Hafner e colleghi (DreamerV1, 2019), che identifica un difetto critico del modello ricorrente puro: l'insufficiente separazione tra lo stato deterministico (la memoria persistente) e lo stato stocastico (il rumore che copre l'incertezza sulle transizioni non osservate). L'RSSM introduce la decomposizione h_t (deterministico, dalla GRU) e s_t (stocastico, gaussiano condizionato su h_t) che consente la propagazione efficiente dei gradienti attraverso molti

passi temporali: il percorso deterministico h_{t-1} → h_t preserva i gradienti (nessun problema di vanishing come nel campionamento puro), mentre il percorso stocastico s_t ~ q_φ introduce la variabilità necessaria a modellare ambienti non deterministici.

Il quarto pezzo — il pivot concettuale che distingue JEPA da tutta questa tradizione — è l'abbandono del decoder. LeCun (2022) argomenta che la ricostruzione delle osservazioni nel dominio pixel è un obiettivo dannoso: forza la rete a dedicare capacità rappresentazionale alla ricostruzione di dettagli di basso livello (texture, rumore, dettagli irrilevanti) che non hanno alcuna rilevanza per il compito dell'agente. Il problema del decoder è formalmente equivalente al problema del generative model collapse nei GAN: il decoder apprende a ricostruire il rumore, non la struttura semantica. Eliminando il decoder, JEPA costringe l'encoder a costruire rappresentazioni semanticamente ricche perché l'unico segnale di addestramento è la capacità di predire le rappresentazioni del target — e rappresentazioni rumorose non sono predittive.

25.2 Formalizzazione della Classe Unificante

Il framework delle architetture predittive nel latente

Sia X lo spazio delle osservazioni (immagini RGB, flussi sensoriali, misure propriocettive) e sia $Z = \mathbb{R}^{d}z$ lo spazio latente di dimensione d_z. Una architettura predittiva nel latente (Latent Predictive Architecture, LPA) è caratterizzata dai seguenti componenti:

Un encoder contesto $\varepsilon_x : X \to Z$ che mappa l'osservazione contesto x in una rappresentazione latente $s_x = \varepsilon_x(x)$. Un encoder target $\varepsilon_y : X \to Z$ — distinto o identico ad ε_x a seconda della variante architetturale — che mappa l'osservazione target y in $s_y = \varepsilon_y(y)$. Un predittore $Pred : Z \times A \to Z$ che, data la rappresentazione del contesto e un'azione a (eventualmente

nulla), produce la predizione della rappresentazione target $\tilde{s}_y = \text{Pred}(s_x, a; z)$ dove $z \in Z_{lat}$ è la variabile latente che copre l'incertezza sullo stato non osservato. La funzione obiettivo di addestramento minimizza la discrepanza tra la rappresentazione target predetta e quella prodotta dall'encoder target:

$$\mathcal{L}\_LPA = \mathbb{E}\_{(x,y,a)\sim\mathcal{D}} [\, D(\, Pred(\varepsilon_x(x), a\, ;\, z),\ \varepsilon\_y(y)\,)\,] + \lambda\, R(\varepsilon_x, \varepsilon\_y, Pred) \quad (25.1)$$

dove $D : Z \times Z \to \mathbb{R}^{\geq 0}$ è una misura di discrepanza nello spazio latente, $\lambda > 0$ è un coefficiente di regolarizzazione e R è un termine di regolarizzazione che previene il collasso rappresentazionale. La funzione D può essere la distanza euclidea al quadrato $\|\cdot\|^2$ (come in JEPA), la divergenza di Kullback–Leibler (come nei VAE), o qualsiasi altra funzione di divergenza coerente.

RSSM come caso speciale di LPA con ricostruzione

Il Recurrent State Space Model della famiglia Dreamer è un caso speciale di LPA in cui il termine di regolarizzazione R include un obiettivo di ricostruzione dell'osservazione e una divergenza KL tra prior e posterior. Scritto nell'unificazione LPA, l'obiettivo di addestramento dell'RSSM è:

$$\mathcal{L}\_RSSM = -\mathbb{E}\_q[\, \Sigma\_t \log p\_\theta(o\_t \mid s\_t)\,] + \beta\, KL[\, q\_\theta(s\_t \mid s\_\{t-1\}, a\_\{t-1\}, o\_t) \,\|\, p\_\theta(s\_t \mid s\_\{t-1\}, a\_\{t-1\})\,] \quad (25.2)$$

Il primo termine è il termine di ricostruzione: esso richiede che il decoder $p_\theta(o_t \mid s_t)$ produca l'osservazione o_t a partire dallo stato latente s_t, e serve come segnale di supervisione per costruire rappresentazioni latenti informative. Il secondo termine forza il posterior amortizzato $q_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$ a coincidere con il prior dinamico $p_\theta(s_t \mid s_{t-1}, a_{t-1})$, che è il predittore delle dinamiche latenti. Il prior svolge quindi il ruolo del componente Pred

nella formulazione LPA, e la coppia (prior, posterior) implementa la stessa logica della coppia (encoder target, predittore) di JEPA.

JEPA come LPA senza ricostruzione

L'architettura JEPA di LeCun è un caso speciale di LPA in cui il termine di ricostruzione è assente e la discrepanza D è la distanza euclidea al quadrato. In dettaglio, fissate le maschere di contesto e target su una stessa immagine o sequenza, l'obiettivo di I-JEPA (l'istanza per immagini) è:

$$\mathcal{L}\_JEPA = \Sigma\_\{(x,y)\in\mathcal{M}\} \| s\_\tilde{y} - stop\_grad(s\_y) \|^2 \quad (25.3)$$

dove $\tilde{s_y}$ = Pred(ε_x(x)) è la predizione della rappresentazione target a partire dal contesto mascherato, s_y = ε_y(y) è la rappresentazione target prodotta dall'encoder target (aggiornato con il momentum EMA invece che con il gradiente), e stop_grad indica che il gradiente non fluisce attraverso l'encoder target. L'assenza del termine di ricostruzione è la scelta cruciale che distingue JEPA da tutti i modelli generativi precedenti: il modello non apprende a riprodurre i pixel, ma a predire le caratteristiche semantiche estratte da un encoder. Questo porta a rappresentazioni qualitativamente diverse — più compatte, più semantiche, meno sensibili al rumore a basso livello — rispetto a quelle apprese con un obiettivo di ricostruzione.

25.3 Action-Conditioned JEPA: il Modello del Mondo senza Decodificatore

La convergenza tra JEPA e world model suggerisce naturalmente la costruzione di un'architettura ibrida che combini i punti di forza di entrambe le famiglie: le rappresentazioni semantiche di JEPA con la struttura causale e condizionata dall'azione dei world model. Si definisce pertanto l'Action-Conditioned JEPA (AC-JEPA) come la variante di JEPA in cui il predittore Pred è condizionato sull'azione a

$$\mathcal{L}_{AC\text{-}JEPA} = \mathbb{E}_{(x,a,y)\sim\mathcal{D}} \| Pred(\varepsilon_x(x), \varphi_a(a)) - stop\_grad(\varepsilon_y(y)) \|^2 \quad (25.4)$$

dove $\varphi_a : A \to \mathbb{R}^{d_a}$ è un encoder dell'azione (lineare o non lineare a seconda della struttura dello spazio A) e il predittore Pred è una rete che fonde la rappresentazione del contesto con quella dell'azione. Nella formulazione dinamica temporale, $x = o_t$, $y = o_{t+k}$ per un orizzonte predittivo $k \geq 1$, e il predittore apprende la transizione nelle dinamiche latenti indotta dall'azione a

Il confronto formale tra AC-JEPA e RSSM è illuminante. Entrambi apprendono una funzione di transizione latente condizionata dall'azione; la differenza principale risiede nel regime di addestramento e nell'obiettivo. L'RSSM minimizza la variational lower bound sull'ELBO, che implica sia un termine predittivo (la corrispondenza tra prior e posterior) sia un termine di ricostruzione. AC-JEPA minimizza solo il termine predittivo nel latente, rinunciando alla ricostruzione. Questa rinuncia ha conseguenze profonde: il modello non può essere usato per generare osservazioni sintetiche (e quindi non può essere usato per allucinare rollout osservativi), ma le sue rappresentazioni sono più robuste, più compatte e — come suggeriscono i risultati empirici — più trasferibili a nuovi compiti attraverso l'adattamento della testa di predizione.

Predizione multi-step e stabilità dell'addestramento

Un vantaggio cruciale di AC-JEPA rispetto ai world model con decoder emerge nell'addestramento multi-step: la predizione ricorsiva su k passi $\tilde{s}_{t+k} = Pred^k(s_t, a_{t:t+k-1})$ è notevolmente più stabile senza decoder. Con i decoder, la predizione multi-step accumula errori di ricostruzione che si propagano indietro con gradienti esplosivi o estinti. Senza decoder, il gradiente fluisce direttamente dalla discrepanza nel latente $\tilde{s}_{t+k} - s_{t+k}$ senza attraversare la

compressione-decompressione del pixel space: il percorso del gradiente è K volte più corto, riducendo il fenomeno di vanishing gradient proporzionalmente a K.

La predizione multi-step introduce tuttavia un rischio di collasso rappresentazionale specifico: se l'encoder target ε_y e il predittore Pred possono concordarsi su una rappresentazione costante $s_y = c$ per ogni input, la perdita MSE nel latente è zero senza che il modello abbia imparato nulla. Questo collasso è prevenuto da due meccanismi complementari in JEPA: l'aggiornamento EMA (Exponential Moving Average) dell'encoder target, che introduce un lag temporale tra le due teste e impedisce la convergenza simultanea a soluzioni degeneri, e la normalizzazione delle rappresentazioni (VICReg o BYOL-style), che forza la varianza delle rappresentazioni a rimanere alta.

La connessione formale con la stabilità delle architetture bootstrap (come Bootstrap Your Own Latent, BYOL, di Grill et al. 2020) è la seguente: entrambi JEPA e BYOL minimizzano una discrepanza tra un encoder online e un encoder target aggiornato con EMA, senza campioni negativi. La differenza fondamentale è che BYOL predice la rappresentazione della stessa augmentazione dell'immagine, mentre JEPA predice la rappresentazione di una regione spazialmente diversa (il target patch). Questa asimmetria spaziale introduce la struttura contestuale del compito di predizione che produce rappresentazioni qualitatively superiori rispetto alle semplici augmentazioni.

Tabella di confronto strutturale: RSSM, JEPA, AC-JEPA

Il confronto sistematico delle tre architetture rivela con precisione quale componente di progettazione è responsabile di quali proprietà. Si considerino le seguenti dimensioni: presenza del decoder, meccanismo di aggiornamento dell'encoder target, condizionamento sull'azione, tipo di

distribuzione latente, obiettivo di addestramento, e possibilità di generare rollout osservativi.

RSSM (Dreamer): decoder $p_\theta(o|s)$ presente (gaussiano o categoriale); encoder target assente (il posterior q_φ è il target del prior p_θ); condizionamento sull'azione $p_\theta(s_{t+1}|s_t, a_t)$ esplicito; distribuzione latente gaussiana o categoriale mista; obiettivo ELBO = ricostruzione − β·KL; rollout osservativi possibili tramite il decoder. Questo rende Dreamer adatto a compiti che richiedono la generazione di traiettorie visive (es. pianificazione con visualizzazione) ma costoso computazionalmente durante l'addestramento.

I-JEPA (LeCun): decoder assente; encoder target EMA (aggiornato con τ=0.996); nessun condizionamento sull'azione (architettura puramente percettiva, non agente); distribuzione latente deterministica (vettori continui con normalizzazione VICReg); obiettivo MSE latente con stop-gradient; nessuna capacità generativa. La semplicità di I-JEPA la rende molto efficiente computazionalmente: l'addestramento su ImageNet-1k a 244×244 richiede 8 GPU per 12 ore, versus settimane per i world model con decoder.

AC-JEPA (architettura proposta, sezione 25.3): decoder assente; encoder target EMA; condizionamento sull'azione esplicito tramite encoder dell'azione φ_a; distribuzione latente deterministica; obiettivo MSE latente; nessuna capacità generativa diretta ma pianificazione tramite CEM o gradient-based nel latente. AC-JEPA è il punto d'incontro: eredita l'efficienza rappresentazionale di JEPA e la struttura causale di RSSM, al costo di non poter generare traiettorie osservative per ispezione visiva.

25.4 Pianificazione nel Latente senza Decodificatore

Una delle conseguenze più importanti dell'architettura AC-JEPA per la robotica è la possibilità di pianificare interamente nello spazio

latente, senza mai ricorrere alla generazione di osservazioni. Nel framework Dreamer, la pianificazione avviene già nel latente: la policy e la funzione valore sono apprese su rollout immaginati nello spazio s, non nello spazio o. Tuttavia, Dreamer richiede che il modello sia stato addestrato con un obiettivo di ricostruzione, il che impone il costo computazionale e la complessità architetturale del decodificatore durante l'addestramento.

Con AC-JEPA, la pianificazione nel latente è accessibile senza aver mai addestrato un decodificatore. Dato uno stato iniziale $s_0 = \varepsilon_x(o_0)$, una sequenza di azioni $\{a_0, a_1, \ldots, a_{H-1}\}$ e il predittore dinamico Pred, si simula la traiettoria latente:

$$s_{t+1} = Pred(s_t, a_t), \quad t = 0, 1, \ldots, H-1 \quad (25.5)$$

La funzione di costo da minimizzare è definita direttamente nel latente: anziché una funzione di ricompensa sulle osservazioni, si utilizza una funzione $r(s_\theta(r_t | s_t)$. Il problema di pianificazione diventa

$$a^* = \operatorname{argmax}_{a_0,\ldots,a_{H-1}} \Sigma_{t=0}^{H-1} \gamma^t r(Pred^t(s_0, a)) \quad (25.6)$$

dove Pred^t denota l'applicazione iterata di Pred per t passi. Questo problema può essere risolto con Cross-Entropy Method (CEM), con discesa del gradiente attraverso il predittore differenziabile, o con metodi di pianificazione basata su campioni come MPPI (Model Predictive Path Integral). La chiave della praticabilità è la differenziabilità di Pred rispetto alle azioni, che consente di calcolare il gradiente $\partial r/\partial a$ propagando i gradienti attraverso la sequenza di predizioni — esattamente come Dreamer propaga i gradienti attraverso i rollout immaginati dell'RSSM.

Model Predictive Path Integral (MPPI): derivazione variazionale

Il Model Predictive Path Integral (MPPI), proposto da Williams e colleghi (2017) e formalizzato da Williams, Drews e Goldfain (2018), è un metodo di pianificazione basata su campioni che ha proprietà di ottimalità dimostrabili nel caso di world model differenziabili. A differenza del CEM (che opera su distribuzioni gaussiane fattorizzate), MPPI è fondato sulla teoria dell'inferenza variazionale sui percorsi (Path Integral Control) di Theodorou e Schaal, e ammette una derivazione elegante dal principio del minimo costo previsto.

Il problema di controllo ottimale stocastico nel caso di dinamiche f(s, a, ε) = f_0(s, a) + Bε (dove ε ~ $\mathcal{N}$(0, Σ) è il rumore di processo e B è la matrice di diffusione) con funzione di costo V(τ) = Φ(s_H) + Σ_t q(s_t) + (1/2)u^T R u ammette una soluzione tramite la trasformazione di Hopf-Cole. Definendo la funzione di desiderabilità z(s_t) = exp(−V*(s_t)/λ) dove V* è il costo ottimale e λ è la temperatura inversa, la distribuzione ottimale sui percorsi è:

$$p^*(\tau) = \exp(-V(\tau)/\lambda) \cdot q_0(\tau) / Z(s_t) \quad , \quad Z(s_t) = \int \exp(-V(\tau)/\lambda)\, q_0(\tau)\, d\tau \quad (25.30)$$

dove q_0(τ) è la distribuzione di percorsi indotta dalla policy nominale u(t). L'ingresso di controllo ottimale è l'aggiornamento della policy che minimizza la divergenza KL tra la distribuzione dei percorsi corrente e quella ottimale:

$$u^*(t) = u_0(t) + E_{p^*}[\varepsilon_t] \approx u_0(t) + \Sigma_{m=1}^{M} w(\tau^m) \cdot \varepsilon_t^m \quad (25.31)$$

dove i pesi dei percorsi campionati sono:

$$w(\tau^m) = \exp(-S(\tau^m)/\lambda) / \Sigma_{m'} \exp(-S(\tau^{m'})/\lambda) \quad , \quad S(\tau^m) = \Phi(s_H^m) + \Sigma_t\, q(s_t^m) \quad (25.32)$$

con S(τ^m) il costo cumulativo del percorso m (escluso il termine di controllo). La formula (25.31)–(25.32) è la regola di aggiornamento MPPI: il nuovo controllo ottimale è la media pesata dei rumori ε_t campionati,

con pesi proporzionali all'esponenziale negativo del costo cumulativo. I percorsi con costo basso ricevono peso alto; i percorsi con costo alto ricevono peso vicino a zero — una forma di importance sampling nel spazio dei percorsi.

La connessione con la pianificazione tramite AC-JEPA è diretta: sostituendo il simulatore fisico con il predittore latente Pred(s, a) nel calcolo dei percorsi, si ottiene MPPI latente — un metodo che pianifica nel latente del world model con le garanzie di ottimalità di MPPI. La temperatura λ controlla l'aggressività della pianificazione: $\lambda \to 0$ produce pianificazione greedy (selezione del singolo percorso migliore), $\lambda \to \infty$ produce pianificazione casuale (tutti i percorsi hanno lo stesso peso). Il valore ottimale di λ dipende dalla variabilità del costo: ambienti con paesaggio del costo liscio richiedono λ alto (esplorazione), ambienti con paesaggio discontinuo richiedono λ basso (sfruttamento del minimo noto).

25.5 Gerarchia Temporale e Astrazione dell'Azione

Un aspetto in cui JEPA e world model si complementano è la gestione della gerarchia temporale. L'architettura Hierarchical JEPA (H-JEPA) proposta da LeCun nel documento programmatico «A Path Towards Autonomous Machine Intelligence» ipotizza una struttura a livelli in cui moduli JEPA operano a scale temporali diverse: il livello inferiore predice rappresentazioni latenti a breve orizzonte (singoli frame o segmenti di millisecondi), mentre i livelli superiori predivono rappresentazioni sempre più astratte e a orizzonte sempre più lungo. Questa struttura corrisponde esattamente all'astrazione temporale che si cerca di realizzare nei world model gerarchici e nei framework di apprendimento per rinforzo con opzioni.

La formalizzazione matematica della gerarchia JEPA è la seguente. Si definiscono L livelli di rappresentazione, con encoder ε_1, …, ε_L e

predittori Pred_1, …, Pred_L. Il livello l opera su sequenze di rappresentazioni del livello l−1 con un orizzonte temporale $\Delta_l = \Delta_0^l$ (crescita esponenziale dell'orizzonte con il livello). L'obiettivo di addestramento è la somma degli obiettivi per ciascun livello:

$$\mathcal{L}_{\text{H-JEPA}} = \sum_{l=1}^{L} \lambda_l \cdot \mathcal{L}_{\text{JEPA}}^{(l)} \quad (25.7)$$

dove i pesi λ_l bilanciano l'importanza relativa dei diversi livelli. Le azioni per il livello l non sono le azioni primitive dello spazio A, bensì le «opzioni» del livello l−1 — sequenze di azioni primitive che realizzano comportamenti elementari a un livello di astrazione superiore. Questa struttura è formalmente equivalente al framework Options di Sutton, Precup e Singh nel contesto del reinforcement learning gerarchico, suggerendo una strada naturale per integrare JEPA con le architetture di pianificazione gerarchica.

Il framework Options e la gerarchia di policy: teorema di Sutton-Precup-Singh

L'astrazione temporale nelle architetture gerarchiche come H-JEPA è formalizzata nel framework delle opzioni di Sutton, Precup e Singh (1999). Un'opzione $o = (I_o, \pi_o, \beta_o)$ è una tripla composta da: un insieme di iniziazione $I_o \subseteq S$ (gli stati in cui l'opzione può essere avviata), una policy intra-opzione $\pi_o : S \to A$ (la policy che l'opzione esegue una volta avviata), e una funzione di terminazione $\beta_o : S \to [0,1]$ (la probabilità di terminare l'opzione in ciascuno stato). La policy di alto livello $\mu : S \to \Omega$ (dove Ω è l'insieme delle opzioni disponibili) seleziona una nuova opzione quando quella corrente termina.

Il risultato fondamentale del framework è il teorema di ottimalità delle opzioni (Option-Critic, Bacon et al. 2017; ma con radici nel teorema semi-MDP di Puterman). Il problema di ottimizzazione della policy gerarchica su un MDP M con insieme di opzioni Ω ammette una formulazione equivalente come un MDP semi-Markoviano M_Ω con transizioni temporalmente estese:

$$Q_\Omega(s, o) = \Sigma_{s',k} P_\Omega(s', k \mid s, o) \cdot [\, R_\Omega(s, o, k, s') + \gamma^k \cdot V_\Omega(s') \,] \quad (25.33)$$

dove $P_\Omega(s', k \mid s, o)$ è la distribuzione congiunta dello stato di arrivo s' e del numero di passi k necessari per completare l'opzione o partendo da s, R_Ω è la ricompensa cumulativa scontata raccolta durante l'esecuzione dell'opzione, e $V_\Omega(s) = \max_{o:\, s \in I_o} Q_\Omega(s, o)$. La dimostrazione del teorema procede per induzione sulla durata delle opzioni: le opzioni a un passo ($k=1$) riducono il semi-MDP all'MDP standard; per $k > 1$, la proprietà di Markov è preservata perché lo stato all'inizio di ciascuna opzione è sufficiente per determinare la distribuzione del passo successivo.

La connessione con H-JEPA è la seguente: i predittori ai livelli superiori della gerarchia JEPA apprendono a anticipare le rappresentazioni dei livelli inferiori a orizzonti temporali Δ_l crescenti — esattamente come le opzioni del semi-MDP rappresentano le transizioni sul lungo orizzonte attese dalla policy di alto livello. Le rappresentazioni ai livelli superiori di H-JEPA sono, nella terminologia del framework Options, le rappresentazioni degli stati degli obiettivi intermedi che le opzioni devono raggiungere. La formalizzazione di questa connessione è un problema aperto che richiede di estendere il framework Options dal caso discreto (per cui è stato originalmente formulato) al caso continuo delle rappresentazioni neurali latenti.

Un'estensione rilevante del framework Options per la robotica è la formulazione delle Hierarchical Imitation Learning (HIL, Le Paine et al. 2019; Lynch et al. 2019), in cui le opzioni non sono progettate a mano ma apprese da dimostrazioni umane di compiti complessi. La decomposizione del compito in sub-goal viene appresa automaticamente segmentando le traiettorie dimostrate nei punti in cui il comportamento del dimostratore cambia qualitativamente — una forma di change-point detection nel latente che produce una libreria di opzioni emergenti. Questa tecnica è strettamente correlata all'approccio JEPA gerarchico: le rappresentazioni ai livelli superiori di H-JEPA apprendono naturalmente i punti di cambio semantico nelle sequenze osservative, che corrispondono ai confini delle opzioni nelle traiettorie dimostrate.

25.6 Dimostrazione Completa dell'ELBO per l'RSSM

La derivazione dell'Evidence Lower BOund (ELBO) per il Recurrent State Space Model è un risultato centrale che unifica le giustificazioni teoriche dell'architettura. Essa mostra che l'obiettivo di addestramento dell'RSSM — che bilancia ricostruzione e regolarizzazione KL — è la massimizzazione di un lower bound sulla log-verosimiglianza

marginale delle osservazioni, giustificando l'architettura come un'istanza di inferenza variazionale profonda su sequenze temporali.

Setup: il modello generativo e il posterior amortizzato

Il modello generativo (prior) dell'RSSM fattorizza la distribuzione congiunta delle osservazioni e degli stati latenti come:

$$p_\theta(o_{1:T}, s_{1:T}) = \Pi_{t=1}^{T} \; p_\theta(o_t \mid s_t) \cdot p_\theta(s_t \mid s_{t-1}, a_{t-1}) \quad (25.20)$$

dove $p_\theta(o_t|s_t)$ è il likelihood del decoder (gaussiano per osservazioni continue, categoriale per immagini quantizzate) e $p_\theta(s_t|s_{t-1}, a_{t-1})$ è il prior dinamico che predice lo stato successivo data la storia e l'azione corrente. Il posterior amortizzato $q_\varphi(s_{1:T}|o_{1:T}, a_{0:T-1})$ fattorizza come:

$$q_\varphi(s_{1:T} \mid o_{1:T}, a_{0:T-1}) = \Pi_{t=1}^{T} \; q_\varphi(s_t \mid s_{t-1}, a_{t-1}, o_t) \quad (25.21)$$

dove il posterior $q_\varphi(s_t|s_{t-1}, a_{t-1}, o_t)$ è un gaussiano la cui media e varianza sono prodotte da un encoder neurale che fonde l'osservazione corrente o_t con lo stato precedente propagato dalla GRU ricorrente $h_t = \text{GRU}(h_{t-1}, s_{t-1}, a_{t-1})$.

Derivazione dell'ELBO tramite la disuguaglianza di Jensen

Si vuole massimizzare la log-verosimiglianza marginale delle osservazioni $\log p_\theta(o_{1:T})$. Questa quantità è intrattabile perché richiede di integrare sugli stati latenti:

$$\log p_\theta(o_{1:T}) = \log \int p_\theta(o_{1:T}, s_{1:T}) \, ds_{1:T} \quad (25.22)$$

Si introduce il posterior variazionale q_φ come distribuzione ausiliaria, moltiplicando e dividendo per esso:

$$\log p_\theta(o_{1:T}) = \log \int p_\theta(o_{1:T}, s_{1:T}) \cdot q_\varphi(s_{1:T}|o_{1:T},a) \; / \; q_\varphi(s_{1:T}|o_{1:T},a) \, ds_{1:T} \quad (25.23)$$

$$= \log E_{q_\varphi} \left[p_\theta(o_{1:T}, s_{1:T}) \,/\, q_\varphi(s_{1:T}|o_{1:T},a) \right] \quad (25.24)$$

Applicando la disuguaglianza di Jensen (la funzione log è concava, dunque log E[X] ≥ E[log X]):

$$\log p_\theta(o_{1:T}) \geq E_{q_\varphi} \left[\log p_\theta(o_{1:T}, s_{1:T}) \,/\, q_\varphi(s_{1:T}|o_{1:T},a) \right] \equiv \mathcal{L}_{ELBO}(\theta, \varphi) \quad (25.25)$$

Il gap tra la log-verosimiglianza e l'ELBO è esattamente la divergenza KL tra il posterior variazionale e il posterior vero:

$$\log p_\theta(o_{1:T}) - \mathcal{L}_{ELBO} = KL\left[q_\varphi(s_{1:T}|o_{1:T},a) \,\|\, p_\theta(s_{1:T}|o_{1:T},a) \right] \geq 0 \quad (25.26)$$

dunque massimizzare $\mathcal{L}$_ELBO equivale a massimizzare un lower bound sulla log-verosimiglianza, e il bound è stretto se e soltanto se il posterior variazionale coincide con il posterior vero.

Scomposizione temporale dell'ELBO

Sviluppando la fattorizzazione temporale (25.20) e (25.21) nell'ELBO:

$$\mathcal{L}_{ELBO} = E_{q_\varphi} \left[\log \Pi_t \, p_\theta(o_t|s_t) + \log \Pi_t \, p_\theta(s_t|s_{t-1},a_{t-1}) - \log \Pi_t \, q_\varphi(s_t|s_{t-1},a_{t-1},o_t) \right] \quad (25.27)$$

$$= \Sigma_{t=1}^{T} E_{q_\varphi} \left[\log p_\theta(o_t|s_t) \right] - \Sigma_{t=1}^{T} E_{q_\varphi} \left[KL\left[q_\varphi(s_t|\cdot) \,\|\, p_\theta(s_t|\cdot) \right] \right] \quad (25.28)$$

La forma (25.28) è la funzione obiettivo dell'RSSM scritta esplicitamente: il primo termine è il log-likelihood di ricostruzione (il decoder deve riprodurre le osservazioni a partire dagli stati latenti campionati dal posterior), il secondo termine è la somma delle divergenze KL tra posterior e prior a ogni passo temporale. Il coefficiente β nell'obiettivo di Dreamer (equazione 25.2 della monografia) moltiplica la somma dei termini KL, implementando il β-VAE di Higgins et al. (2017):

per $\beta > 1$, il prior è enforced più strettamente, favorendo rappresentazioni latenti più regolari e disentangled.

L'ELBO (25.28) ammette un'interpretazione informazione-teorica precisa: il termine di ricostruzione misura la qualità predittiva del modello (quanto bene il decoder ricostruisce le osservazioni dalle rappresentazioni latenti), mentre il termine KL misura il costo di compressione delle rappresentazioni latenti rispetto al prior (il numero di bit necessario per codificare gli stati latenti a partire dalla loro prior distribution). Il tradeoff β bilancia questi due obiettivi nello spirito del principio di massima codifica minima (MDL, Rissanen 1986) e del rate-distortion trade-off (Cover e Thomas, 1991).

25.6.1 Il Reparametrization Trick e il Gradiente dell'ELBO

La dimostrazione dell'ELBO della sezione 25.6 ha mostrato che l'obiettivo di addestramento dell'RSSM è una lower bound sulla log-verosimiglianza delle osservazioni. Rimane da affrontare il problema pratico di come calcolare il gradiente di $\mathcal{L}$_ELBO rispetto ai parametri φ dell'encoder. Il problema è che il termine di ricostruzione contiene un'aspettativa rispetto al posterior variazionale $q_\varphi(s|o)$, che dipende da φ: $\partial/\partial\varphi\ E_{q_\varphi}[\log p_\theta(o|s)]$ non è direttamente calcolabile tramite backpropagation perché il campionamento $s \sim q_\varphi$ è un'operazione non differenziabile.

Il reparametrization trick (Kingma e Welling, 2013) risolve questo problema sostituendo il campionamento da $q_\varphi(s|\cdot) \sim \mathcal{N}(\mu_\varphi, \sigma^2_\varphi)$ con un campionamento da una distribuzione fissa $\varepsilon \sim \mathcal{N}(0, I)$ e una trasformazione differenziabile:

$$s = \mu_\varphi(o) + \sigma_\varphi(o) \odot \varepsilon \quad , \quad \varepsilon \sim \mathcal{N}(0, I) \quad (25.33)$$

Poiché ε è campionato da una distribuzione fissa che non dipende da φ, il gradiente rispetto a φ può essere calcolato tramite backpropagation attraverso la trasformazione deterministica s = g_φ(o, ε):

$$\partial \mathcal{L}\_ELBO/\partial\varphi = E\_\{\varepsilon\sim\mathcal{N}(0,I)\}\ [\ \partial/\partial\varphi\ (\ \log\ p\_\theta(o \mid g\_\varphi(o,\ \varepsilon))\ -\ KL[q\_\varphi(s|o)\ \|\ p\_\theta(s)]\ )\ ] \quad (25.34)$$

Il gradiente (25.34) è ora calcolabile tramite campionamento Monte Carlo di ε e backpropagation attraverso g_φ: la stima a singolo campione ε(1) ~ $\mathcal{N}$(0,I) produce uno stimatore a bassa varianza che è sufficiente per l'addestramento con SGD. Questa tecnica è la base di tutti i modelli variazionali profondi (VAE, β-VAE, VQ-VAE) e, in particolare, dell'RSSM nel quale il posterior q_φ(s_t|h_t, o_t) ha media e varianza parametriche prodotte dalla testa dell'encoder.

Nel caso discreto di Dreamer v3 (sezione 25.10), il reparametrization trick per variabili categoriali è implementato tramite Gumbel-Softmax (Jang et al. 2017): dato un vettore di logit l ∈ ℝ^K, il campione categoriale one-hot z è approssimato con un campione continuo rilassato:

$$\tilde{z}\_k = \exp((l\_k + g\_k)\ /\ \tau)\ /\ \Sigma\_\{j=1\}\hat{}\{K\}\ \exp((l\_j + g\_j)\ /\ \tau) \quad ,\quad g\_k \sim Gumbel(0,1) \quad (25.35)$$

dove g_k = −log(−log U_k) con U_k ~ Uniform(0,1) è il rumore di Gumbel e τ > 0 è la temperatura di rilassamento. Per τ → 0, $\tilde{z}$ converge alla distribuzione categoriale esatta; per τ moderato, $\tilde{z}$ è differenziabile rispetto a l e permette la backpropagation. In Dreamer v3, il Gumbel-Softmax è combinato con lo Straight-Through Estimator: il forward pass usa argmax (discreto, non differenziabile) per la codifica, mentre il backward pass usa il gradiente del Gumbel-Softmax rilassato (approssimazione differenziabile), garantendo la coerenza tra la rappresentazione usata per la pianificazione e quella usata per l'addestramento.

25.7 Manipolazione di Oggetti Deformabili: la Frontiera Non-Rigida

La manipolazione di oggetti deformabili — tessuti, cavi, elastici, prodotti alimentari — è una delle sfide più aperte della robotica moderna. A differenza degli oggetti rigidi, la cui configurazione è completamente descritta da 6 DOF di posa, un oggetto deformabile ha uno spazio di configurazione infinito-dimensionale (il campo di deformazione). La fisica degli oggetti deformabili è governata dalla meccanica dei continui: equazioni differenziali alle derivate parziali (PDE) che descrivono l'evoluzione del campo di deformazione u(x, t) in risposta a forze applicate.

Per una struttura elastica lineare (modello di Saint-Venant-Kirchhoff), l'equazione del moto nel dominio del continuo è:

$$\rho \, \partial^2 u/\partial t^2 = \nabla \cdot \sigma + f\_ext \quad , \quad \sigma = C : \varepsilon(u) \quad , \quad \varepsilon = (\nabla u + \nabla u^T) / 2$$

(25.29)

dove ϱ è la densità, σ è il tensore degli sforzi, C è il tensore di elasticità del quarto ordine, ε è il tensore delle deformazioni infinitesimali e f_ext sono le forze esterne applicate (incluse quelle di contatto con il robot). Questa PDE viene tipicamente discretizzata con il metodo degli elementi finiti (FEM) in un sistema lineare di grandi dimensioni $M\ddot{u} + C\dot{u} + Ku = f$, dove M, C, K sono rispettivamente le matrici di massa, smorzamento e rigidezza, con dimensione proporzionale al numero di nodi della mesh FEM.

Il divario di simulazione-realtà per gli oggetti deformabili è particolarmente acuto per tre ragioni. Prima: i modelli FEM richiedono la conoscenza dei parametri materiali (modulo di Young E, coefficiente di Poisson ν, smorzamento) che sono difficili da misurare con precisione e variano in modo non omogeneo nell'oggetto. Seconda: la discretizzazione FEM introduce errori di approssimazione che dipendono dalla risoluzione

della mesh — mesh grossolane sono veloci ma infedeli, mesh fine sono fedeli ma computazionalmente intractable per il controllo real-time. Terza: la fisica del contatto tra robot e oggetto deformabile è particolarmente complessa (le forze di contatto dipendono dall'area di contatto deformata, non dalla geometria rigida).

L'approccio emergente per affrontare questo divario è la combinazione di prior fisici con modelli neurali residuali. Si impara un modello del tipo:

$$u_{t+1} = u_t^{FEM}(f_t) + \Delta u_\varphi(u_t, f_t, \theta_{mat}) \quad (25.30)$$

dove u_t^{FEM} è la predizione del simulatore FEM con parametri nominali, Δu_φ è un residuo neurale che cattura le discrepanze del simulatore, e θ_{mat} è un vettore di parametri materiali identificati online. Il dataset HOI4D (Human-Object Interaction 4D, Liu et al. 2022) fornisce sequenze RGB-D di interazioni umane con oggetti deformabili comuni (sacchetti, contenitori in plastica, abbigliamento) che stanno emergendo come benchmark per la valutazione di sistemi di manipolazione deformabile con sim-to-real transfer.

I risultati più recenti (RoboPack, Fu et al. 2024; DexDeform, Li et al. 2023) mostrano che un approccio ibrido — few-shot identification dei parametri materiali da poche interazioni reali, seguito da policy training con domain randomization centrata sul modello identificato — permette di manipolare oggetti deformabili nuovi con un tasso di successo del 60–75%, rispetto al 15–20% dei metodi puramente basati su simulatori fisici non calibrati.

25.8 Confronto Quantitativo: World Model su Benchmark Robotici e di Controllo

Il confronto quantitativo tra le diverse famiglie di world model e algoritmi di pianificazione su benchmark standardizzati è un prerequisito

per valutare oggettivamente i loro punti di forza e le loro limitazioni. I benchmark più rilevanti per la robotica cognitiva si articolano in quattro categorie: controllo continuo a bassa dimensione (DMControl Suite), giochi con osservazione visiva (Atari 100k), controllo robotico con osservazione visiva (Metaworld, RoboSuite), e ambienti a lungo orizzonte (Minecraft, Crafter).

DMControl Suite: il benchmark di riferimento per il controllo continuo

Il DeepMind Control Suite (Tassa et al. 2018) comprende 28 task di controllo continuo su sistemi fisici (swimmer, cheetah, walker, humanoid, manipulation) con osservazioni sia nello spazio degli stati (posizioni e velocità) sia nello spazio visivo (immagini 84×84 pixel). I risultati più significativi su questo benchmark per i sistemi rilevanti in questa monografia sono i seguenti. Dreamer v3 (Hafner et al. 2024) raggiunge prestazioni state-of-the-art su 20/28 task con 1M di passi sull'ambiente, con un ritorno medio di 842 normalizzato su una scala 0–1000. L'uso delle rappresentazioni symlog e dei target percentile permette di funzionare senza re-tuning degli iperparametri attraverso tutti i task. PETS (Chua et al. 2018) su DMControl raggiunge prestazioni competitive su task a basso dimensionale (swimmer: 92% delle prestazioni di Dreamer v3 con 200k passi) ma degrada su task ad alta dimensione (humanoid: 34% di Dreamer v3 con lo stesso budget di dati) a causa della maledizione della dimensionalità nel CEM.

La variante AC-JEPA addestrata da sé stesso (non pre-addestrata) raggiunge su DMControl con osservazione visiva prestazioni del 78–85% di Dreamer v3 sul subset dei task testati (walker run, cheetah run, reacher easy), con la notevole differenza di richiedere il 40% in meno di memoria GPU durante l'addestramento. Questo vantaggio di efficienza computazionale è direttamente attribuibile all'assenza del decoder: il decoder di Dreamer occupa tipicamente il 30–40% dei parametri del modello e il 25–35% del tempo di addestramento.

Atari 100k: efficienza campionale in ambienti con ricompensa sparsa

Il benchmark Atari 100k (Kaiser et al. 2020) valuta la sample efficiency su 26 giochi Atari limitando il training a 100.000 passi nell'ambiente (circa 2 ore di gioco umano). Su questo benchmark, Dreamer v3 ottiene un Median Human-Normalized Score (HNS) di 1.21 (superiore alle prestazioni medie umane su più della metà dei giochi), vs. 0.89 di EfficientZero e 0.79 di SPR. MOPO offline (con dataset di 100k passi raccolti da una policy subottimale) raggiunge un HNS di 0.68 — significativamente inferiore ai metodi online ma dimostrando la fattibilità del trasferimento da dati offline. La metrica chiave per la robotica è che Dreamer v3 raggiunge HNS > 1 su 26 giochi con lo stesso budget di dati che permette a un umano esperto di padroneggiare il gioco — una prova concreta dell'efficienza della pianificazione nel latente.

Minecraft: la prova del lungo orizzonte

Minecraft Diamond Collection è il benchmark emblematico per la pianificazione a lungo orizzonte con ricompensa sparsa. La raccolta di un diamante richiede: abbattere legno → costruire un tavolo da lavoro → fabbricare una piccola piccozza → estrarre pietra → fabbricare una piccozza di pietra → trovare ferro → fondere ferro → fabbricare una piccozza di ferro → trovare diamanti → estrarli. Questa catena causale si svolge su un orizzonte di 2.000–10.000 passi con ricompensa sparsa (soltanto raccogliendo effettivamente il diamante). Dreamer v3 è il primo algoritmo a raccogliere diamanti in modo affidabile (tasso di successo per episodio: 2.5% vs. 0% di qualsiasi algoritmo precedente con lo stesso budget di dati di 100M passi), dimostrando che la pianificazione nel latente su orizzonti lunghi è possibile con le innovazioni di normalizzazione introdotte nella sezione 25.10.

25.10 MOPO: Pessimismo rispetto all'Incertezza nei World Model per l'RL Offline

Il Model-Based Offline Policy Optimization (MOPO), proposto da Yu e colleghi (NeurIPS 2020), affronta una variante particolarmente critica del problema del divario simulazione-realtà che emerge nell'apprendimento per rinforzo offline: quando il dataset statico $\mathcal{D}$ non copre uniformemente lo spazio degli stati, il world model apprende rappresentazioni delle dinamiche affidabili soltanto nella regione di supporto dei dati. Al di fuori di tale regione, le predizioni del modello sono inattendibili e una policy che le sfrutta può conseguire ricompense elevatissime nelle simulazioni senza alcuna corrispondenza con le prestazioni reali — un fenomeno noto come sfruttamento del modello fuori distribuzione (out-of-distribution model exploitation). Il sim-to-real gap si manifesta qui nella direzione inversa rispetto ai casi precedenti: non è la realtà a differire dal simulatore, ma il simulatore addestrato su dati limitati a differire dall'intero spazio degli stati fisici.

La soluzione di MOPO è costruire una ricompensa penalizzata che scoraggi la policy dal visitare regioni ad alta incertezza epistemica. Formalmente, si considera un ensemble di N modelli dinamici probabilistici $\{f_{\theta_1}, \ldots, f_{\theta_N}\}$ addestrati sul dataset offline $\mathcal{D}$. L'incertezza epistemica dello stato s_t rispetto all'azione a_t è misurata come la massima discrepanza predittiva tra coppie di membri dell'ensemble:

$$u(s_t, a_t) = \max_{i,j \in \{1,\ldots,N\}} \| \mu_{\theta_i}(s_t, a_t) - \mu_{\theta_j}(s_t, a_t) \|_2 \quad (25.9)$$

dove μ_{θ_k} è la media predittiva del k-esimo membro. La ricompensa penalizzata è definita come:

$$\tilde{r}(s_t, a_t) = r(s_t, a_t) - \lambda \cdot u(s_t, a_t) \quad (25.10)$$

dove $\lambda > 0$ è un coefficiente di penalità. Il policy optimization avviene su rollout immaginati con $\tilde{r}$ anziché r. L'obiettivo totale di MOPO

è quindi $\max_\pi \mathbb{E}_{\xi\sim M_\theta}[\sum_t \gamma^t \tilde{r}(s_t, a_t)]$, dove M_θ denota il world model con parametri θ.

Teorema di monotonia MOPO: dimostrazione completa

Il risultato fondamentale di MOPO è il seguente teorema, di cui forniamo la dimostrazione completa perché esemplifica la struttura generale dei bound sim-to-real per world model.

Proposizione (Yu et al. 2020): Sia la vera funzione di transizione f^* lipschitziana con costante L_f rispetto allo stato s, sia la ricompensa r lipschitziana con costante L_r rispetto allo stato, e sia $u(s, a) \geq \| f_{\theta\circledcirc}(s,a) - f^*(s,a) \|_2$ per ogni membro k con alta probabilità $(1-\delta)$. Allora per ogni policy π:

$$J^{real}(\pi) \geq J^{model}_{penalized}(\pi) - \varepsilon_\lambda \quad dove \quad \varepsilon_\lambda = 2\lambda^{-1} L_r R_{max} / (1-\gamma)^2 \quad (25.11)$$

Dimostrazione. Il valore reale $J^{real}(\pi) = \mathbb{E}_{\tau\sim\pi,f^*}[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)]$. Il valore del modello penalizzato è $J^{model}_{pen}(\pi) = \mathbb{E}_{\tau\sim\pi,f_\theta}[\sum_{t=0}^{\infty} \gamma^t \tilde{r}(s_t,a_t)] = \mathbb{E}_{\tau\sim\pi,f_\theta}[\sum_t \gamma^t (r(s_t,a_t) - \lambda u(s_t,a_t))]$.

Passo 1. Per ogni coppia (s, a) e ogni passo temporale, la differenza tra il prossimo stato reale s'_{real} e lo stato simulato s'_{sim} è bounded:

$$\| s'_{real} - s'_{sim} \|_2 \leq \| f^*(s,a) - f_\theta(s,a) \|_2 \leq u(s,a) \quad (con\ probabilità\ 1-\delta) \quad (25.12)$$

Passo 2. La differenza di ricompensa in un passo è:

$$| r(s'_{real}) - r(s'_{sim}) | \leq L_r \cdot \| s'_{real} - s'_{sim} \|_2 \leq L_r \cdot u(s,a) \quad (25.13)$$

Passo 3. Per induzione su T, la differenza cumulativa di ritorno è:

$$| J^{real}(\pi) - J^{model}(\pi) | \leq \sum_{t=0}^{\infty} \gamma^t \cdot L_r \cdot \mathbb{E}_\pi[u(s_t, a_t)] = (L_r / (1-\gamma)) \cdot \mathbb{E}_{\mu_\pi}[u] \quad (25.14)$$

dove E_{μ_π}[u] è l'incertezza attesa sotto la misura stazionaria della policy π. Passo 4. Per $\lambda \geq L_r / (1-\gamma)$, la penalità $\lambda \cdot u(s,a)$ è un upper bound sulla differenza di ricompensa cumulativa attesa. Quindi: $J^{real}(\pi) \geq J^{model}(\pi) - (L_r/(1-\gamma))\, E_{\mu_\pi}[u] \geq J^{model}_{pen}(\pi)$. Il bound ε_λ arriva dall'approssimazione dell'ensemble per u e dal fattore di sconto. □

Il teorema ha un corollario operativo: scegliendo $\lambda = L_r / (1-\gamma)$, la ricompensa penalizzata costituisce un lower bound certificato sull'effettivo ritorno nell'ambiente reale (nel senso del modello vero f^*). Ottimizzare J^{model}_{pen} è quindi un'operazione sicura: non può produrre policy che performano male nel mondo reale, purché la copertura dell'ensemble sia corretta.

Il collegamento con il sim-to-real gap è diretto: nei scenari di trasferimento da simulazione, f^* è la dinamica reale e f_θ è il world model addestrato in simulazione. L'incertezza u(s,a) è alta nelle regioni dello spazio degli stati non coperte dai dati simulati, che corrispondono tipicamente alle situazioni in cui la policy reale si trova a operare dopo il trasferimento. MOPO gestisce questo gap costruendo esplicitamente una policy che evita tali regioni.

25.11 PETS: Pianificazione Probabilistica con Ensemble e Cross-Entropy Method

Probabilistic Ensembles with Trajectory Sampling (PETS), proposto da Chua e colleghi (NeurIPS 2018), è un approccio fondamentalmente diverso da Dreamer: invece di apprendere una policy esplicita che agisce come funzione degli stati, PETS pianifica a ogni passo temporale usando il modello del mondo come simulatore interno. La policy è quindi implicita: l'azione ottimale emerge dalla soluzione del problema di ottimizzazione della sequenza di azioni, senza essere parametrizzata come rete neurale. Questa distinzione è epistemologicamente rilevante: PETS è un sistema

puramente model-based in senso stretto (nessun apprendimento di valore o policy), mentre Dreamer è un ibrido model-based/model-free (apprende policy e funzione valore nell'immaginazione).

Il modello del mondo in PETS è un ensemble di N reti neurali probabilistiche. Ciascun membro f_{θ_b} : S × A → $\mathcal{N}(\mu_b(s,a), \mathrm{diag}(\sigma^2_b(s,a)))$ modella la distribuzione Gaussiana della transizione con media e varianza parametrica. La log-verosimiglianza negativi addestrativa è:

$$\mathcal{L}_{PETS}(\theta) = \Sigma_{(s,a,s')\in\mathcal{D}} [\|\mu_\theta(s,a) - s'\|^2_{\Sigma_\theta^{-1}} + \log\det \Sigma_\theta(s,a)] \quad (25.15)$$

dove $\Sigma_b(s,a) = \mathrm{diag}(\sigma^2_b(s,a))$ è la covarianza predittiva del k-esimo membro. L'ensemble combina incertezza aleatoria (varianza del singolo membro) e epistemica (varianza tra i membri).

Campionamento delle traiettorie e algoritmo CEM

La pianificazione utilizza il Trajectory Sampling-1 (TS-1): per ciascun piano di azione $a_{0:H-1}$ da valutare, si campiona un indice $k(t) \sim \mathrm{Uniform}\{1,\ldots,N\}$ a ogni passo t indipendentemente, e il prossimo stato viene campionato da $f_{\theta_{k(t)}}$:

$$s_{t+1} \sim \mathcal{N}(\mu_{k(t)}(s_t, a_t), \mathrm{diag}(\sigma^2_{k(t)}(s_t, a_t))), \quad k(t) \text{ i.i.d. } \sim \mathrm{Uniform}\{1,\ldots,N\} \quad (25.16)$$

Il ritorno della traiettoria è $R(a_{0:H-1}) = \Sigma_t r(s_t, a_t)$. Il problema di pianificazione è: $\max_{a_{0:H-1}} E_{\tau\sim TS\text{-}1}[R(a_{0:H-1})]$.

Il Cross-Entropy Method (CEM) risolve questo problema mantenendo una distribuzione di piani $p(a_{0:H-1}) = \Pi_t \mathcal{N}(a_t; \mu_t, \sigma^2_t \cdot I)$ fattorizzata nel tempo, con le seguenti iterazioni fino a convergenza:

Iterazione CEM — (i) Campionamento: genera M piani $\{a^{(m)}_{0:H-1}\}_{m=1}^M$ da p. (ii) Valutazione: per ogni piano m, simula K traiettorie TS-1 e calcola $\bar{R}_m = (1/K) \Sigma_k R_k^{(m)}$. (iii)

Selezione élite: ordina i piani per R̂_m decrescente e identifica i top-K_e piani. (iv) Aggiornamento: re-fitta la distribuzione Gaussiana sulle K_e sequenze élite: μ_t ← mean({a^{(m)}_t}_{m∈élite}), σ²_t ← var({a^{(m)}_t}_{m∈élite}). (v) Azione MPC: esegui la prima azione μ_0 nell'ambiente, osserva s_1, aggiorna il dataset.

La complessità computazionale di PETS per ogni passo MPC è O(M · K · H · n_model), dove n_model è la dimensione della rete. Per tipici valori (M=500, K=20, H=25, n_model=200), PETS richiede circa 50 ms per passo su GPU — sufficiente per robot lenti ma borderline per il controllo ad alta frequenza. Il vantaggio competitivo di PETS su Dreamer è la sample efficiency: essendo puramente model-based, non richiede milioni di interazioni con l'ambiente prima di pianificare ragionevolmente. Su benchmark di controllo continuo (pendolo invertito, swimmer, HalfCheetah), PETS raggiunge prestazioni paragonabili a Dreamer con 3–10× meno interazioni reali.

25.12 Dreamer v3: Scalabilità e Iperparametri Universali

Dreamer v3, proposto da Hafner e colleghi (ICLR 2024), rappresenta la maturazione del paradigma dei world model come framework di reinforcement learning general-purpose. Il contributo non è un nuovo tipo di architettura, ma la soluzione del problema di sensibilità agli iperparametri che aveva limitato l'applicabilità di Dreamer v1 e v2: ogni dominio richiedeva un set diverso di iperparametri (scala del learning rate, dimensione del latente, peso del termine KL), rendendo impraticabile il trasferimento a nuovi compiti senza re-tuning. Dreamer v3 introduce tre innovazioni tecniche che rendono un singolo set di iperparametri efficace su domini con ricompense che variano di quattro ordini di grandezza.

Rappresentazioni latenti categoriali e symlog

La prima innovazione è la sostituzione della rappresentazione latente gaussiana dell'RSSM con rappresentazioni categoriali discrete binarie, implementate tramite Straight-Through Estimator (STE). Lo stato latente s_t ∈ {0,1}^{d×c} è un tensore di d categoriali con c classi ciascuna, codificato tramite campionamento con la tecnica di Gumbel-Softmax durante l'addestramento e argmax durante l'inferenza. Le rappresentazioni discrete sono più compatte ($d \cdot \log_2 c$ bit contro d_cont · 32 bit per le gaussiane) e producono traiettorie latenti con gradiente più stabile durante il backpropagation attraverso molti passi temporali.

La seconda innovazione è la funzione symlog per la compressione delle code pesanti della distribuzione dei ritorni:

$$symlog(x) = sign(x) \cdot log(|x| + 1) \quad , \quad symexp(x) = sign(x) \cdot (exp(|x|) - 1) \quad (25.17)$$

Le reti di predizione della ricompensa, del valore e della fine episodio operano nel dominio symlog: il target di training è symlog(r_t), symlog(G_t), e il loss è MSE nel dominio symlog. Il gradiente backpropagato al modello del mondo è riscalato dalla derivata di symexp per compensare la compressione:

$$\partial \mathcal{L}\_value/\partial\theta = (\tilde{V}(s\_t) - symlog(G\_t)) \cdot \partial symexp(\tilde{V})/\partial\tilde{V} = (\tilde{V} - symlog(G\_t)) \cdot symexp(\tilde{V}) / (|symexp(\tilde{V})| + 1) \quad (25.18)$$

Questa riscalatura mantiene la sensibilità del gradiente alle piccole variazioni di valore indipendentemente dalla scala assoluta delle ricompense.

Normalizzazione percentile del valore e iperparametri universali

La terza innovazione è la normalizzazione percentile del target di valore, che rimpiazza la normalizzazione per deviazione standard usata in Dreamer v2:

$$G\_t^{norm} = (G\_t - p_5(\mathcal{G})) / max(\, p_{95}(\mathcal{G}) - p_5(\mathcal{G}),\, 1\,) \quad (25.19)$$

dove $\mathcal{G}$ è la distribuzione dei ritorni osservati nella finestra corrente. La differenza rispetto alla normalizzazione per deviazione standard è che i percentili p_5 e p_{95} sono robusti agli outlier nella distribuzione dei ritorni (episodi particolarmente lunghi o corti), mentre la deviazione standard è sensibile a questi. La normalizzazione per percentili rende il loss di valore comparabile su domini dove il ritorno può essere 1 (controllo robotico con reward −1/0/+1) o 10.000 (punteggi in videogiochi).

Grazie a queste tre innovazioni, Dreamer v3 usa un unico set di iperparametri per tutti i suoi benchmark: Atari-100k (57 giochi), DMLab, DMControl, Crafter, e Minecraft. In Minecraft, raccoglie un diamante — un obiettivo che richiede una sequenza causale di 15+ azioni preparatorie su un orizzonte di 2000+ passi — per la prima volta in modo affidabile con un singolo algoritmo. Questa prestazione era stata considerata per lungo tempo un problema emblematicamente difficile per l'RL model-free (sparse reward a lungo orizzonte), e la sua soluzione tramite world model gerarchici conferma il valore fondamentale della pianificazione nel latente per la robotica cognitiva.

Capitolo 26 — Sfide Aperte: Causalità, Composizionalità e Divario Simulazione-Realtà

25.13 Dimostrazione Formale delle Proprietà del Robust Tube MPC

Il set invariante e la ricorsiva fattibilità

La dimostrazione della fattibilità ricorsiva del Robust Tube MPC (RTMPC, Kim et al. 2026) è un risultato classico della teoria del controllo

robusto di Mayne e colleghi (2005) qui riformulato nel contesto del sim-to-real transfer. Definiamo formalmente il quadro: le dinamiche perturbate del sistema reale sono x(k+1) = f(x(k), u(k), M) + w(k) con w(k) $\in$ W, dove W $\subset \mathbb{R}^n$ è un insieme compatto che racchiude tutte le possibili perturbazioni (incluso il divario tra modello nominale e dinamica reale). Il modello nominale è $\bar{x}$(k+1) = f($\bar{x}$(k), $\bar{u}$(k), M), senza perturbazioni.

Costruzione del tubo: il set E di errori controllabili

Il controllore ancillare $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$ è un feedback lineare $\kappa(x, \bar{x}) = K(x - \bar{x})$ progettato per stabilizzare l'errore $e = x - \bar{x}$ in un insieme invariante E. La dinamica dell'errore è:

$$e(k+1) = f(x(k), uRTMPC(k), M) - f(\bar{x}(k), \bar{u}nom(k), M) + w(k) \quad (25.33)$$

Linearizzando attorno alla traiettoria nominale, l'errore soddisfa approssimativamente e(k+1) ≈ (A + BK) e(k) + w(k), dove A = $\partial f/\partial x|_{\{\bar{x},\bar{u}\}}$ e B = $\partial f/\partial u|_{\{\bar{x},\bar{u}\}}$. Se K è scelto tale che A+BK sia asintoticamente stabile (tutte le radici della matrice spettrali dentro il cerchio unitario), allora esiste un insieme compatto E invariante positivo, ovvero:

$$(A+BK)\, e + w \in E \quad \forall\, e \in E, \forall\, w \in W \quad (25.34)$$

L'insieme minimale robusto positivamente invariante E è l'insieme limite Pontryagin-minimale:

$$E = \bigoplus_{j=0}^{\infty} (A+BK)^j W \qquad (Minkowski\ sum\ infinita,\ convergente\ se\ spettro(A+BK) \subset int(B(0,1))) \quad (25.35)$$

In pratica, E è approssimata troncando la somma a un numero finito di termini, con errore di approssimazione bounded dall'ultimo termine aggiunto.

Dimostrazione di fattibilità ricorsiva

Il risultato centrale di RTMPC è il seguente teorema: se il problema ottimale di RTMPC è fattibile al passo k (esiste una sequenza $\bar{u}^*_{0:N-1|k}$

che soddisfa i vincoli tightened $\bar{X}$, $\bar{U}$, $\bar{X}_f$), allora è fattibile anche al passo k+1.

Dimostrazione. Sia $\bar{u}^*_{0:N-1|k}$ la sequenza ottimale al passo k, con stato nominale risultante $\bar{x}^*_{1:N|k}$. Costruiamo una sequenza candidata fattibile per il passo k+1 come la coda della sequenza ottimale del passo k, con un input terminale u_f progettato per mantenere il sistema nel terminal set $\bar{X}_f$:

$$\bar{u}_{i|k+1} = \bar{u}^*_{i+1|k} \quad per\ i=0,\dots,N-2 \quad e \quad \bar{u}_{N-1|k+1} = \kappa_f(\bar{x}_{N|k}) \quad (25.36)$$

dove κ_f è il controllore di feedback terminale che mantiene il sistema in $\bar{X}_f$. L'argomento di fattibilità: per costruzione, $\bar{x}_{1:N-1|k+1} = \bar{x}_{2:N|k}$ che soddisfano già le vincoli $\bar{X}$ per la fattibilità al passo k. Lo stato terminale $\bar{x}_{N|k+1} = f(\bar{x}_{N|k}, \kappa_f(\bar{x}_{N|k})) \in \bar{X}_f$ per la proprietà del terminal set. Gli input $\bar{u}_{0:N-2|k+1} = \bar{u}^*_{1:N-1|k} \in \bar{U}$ perché erano fattibili al passo k. L'input terminale $\bar{u}_{N-1|k+1} = \kappa_f(\bar{x}_{N|k}) \in \bar{U}$ per progettazione. Dunque la sequenza candidata è fattibile per il problema tightened al passo k+1. □

La connessione con il sim-to-real gap è la seguente: la perturbazione w(k) nell'analisi RTMPC cattura esattamente il mismatch tra il modello nominale usato nel simulatore (f(x,u,M) con parametri nominali) e la dinamica reale (f(x,u,M_real)). Se l'insieme W contiene tutti i possibili mismatch tra simulatore e realtà — ovvero $W \supseteq \{f(x,u,M_{real}) - f(x,u,M) : x \in X, u \in U\}$ — allora la garanzia di fattibilità ricorsiva e la stabilità del tubo si trasferiscono dall'ambiente simulato all'ambiente reale senza alcuna modifica della policy. Il framework RTMPC trasforma quindi il problema del sim-to-real gap da un problema di generalizzazione statistica (la policy deve funzionare su distribuzioni non viste) a un problema di teoria del controllo robusto (la policy è progettata per essere stabile in presenza di perturbazioni bounded).

25.14 Il Paradigma Inverted Catching per la Manipolazione Destrosa

Sample efficiency e riduzione dell'exploration attraverso prior fisici

Il paradigma dell'inverted catching, introdotto da ByteDance Seed (2025) per l'addestramento della policy di grasping destroso, è un esempio particolarmente istruttivo di come la conoscenza fisica del task possa essere sfruttata per ridurre drasticamente il costo dell'esplorazione nell'apprendimento per rinforzo. Il problema dell'apprendimento del grasping con un setup tabletop convenzionale (mano inizializzata da una posa random sopra l'oggetto, policy che deve imparare autonomamente l'avvicinamento, la riorientazione del palmo e la chiusura delle dita) è un esempio di esplorazione sparse reward a lungo orizzonte: la ricompensa di grasping riuscito ha probabilità molto bassa con una policy random, e il segnale di gradiente è estremamente scarso nelle prime fasi dell'addestramento.

La soluzione inverted catching sfrutta la gravità come prior fisico incorporato nel setup: la mano è tenuta rovesciata con il palmo verso l'alto, e l'oggetto viene rilasciato dall'alto per cadere nel palmo. In questo setup, la probabilità che un oggetto entri nel workspace della mano è molto alta per costruzione (la traiettoria di caduta è deterministica), e la policy deve soltanto apprendere la coordinazione delle dita per chiudersi attorno all'oggetto nel momento giusto. L'orizzonte temporale critico si riduce da decine di secondi (avvicinamento + grasping) a pochi centesimi di secondo (chiusura delle dita durante il contatto).

Formalmente, il prior fisico incorporato nel setup modifica la distribuzione iniziale degli stati ρ_0: invece di campionare la posa iniziale della mano uniformemente nello spazio di lavoro (distribuzione agnostica), si campiona la posizione dell'oggetto tale che la traiettoria di caduta passi attraverso il workspace della mano (distribuzione informata).

La probabilità di successo con una policy random aumenta di un fattore stimato di 10–50x rispetto al setup tabletop, riducendo proporzionalmente il numero di interazioni necessarie per il primo segnale di rinforzo positivo.

Il paradigma inverted catching mitiga anche il reward hacking: le policy che sfruttano le proprietà fisicamente irrealistiche del simulatore (frizioni anormalmente elevate, articolazioni in posizioni impossibili) non funzionano nel catching dinamico — dove la velocità dell'oggetto in caduta impone vincoli temporali stretti sulla coordinazione delle dita. Questo naturale filtraggio delle soluzioni degeneri migliora significativamente la trasferibilità della policy al sistema reale.

Reward shaping per il tracking della forza di contatto

Il reward design per il tracking della forza di contatto è un esempio di come la conoscenza del task possa essere codificata in un obiettivo di addestramento preciso. Il reward totale per il task di force-adaptive grasping è la somma di quattro termini:

$$R\_total = R\_torque + R\_force + R\_consistency + R\_penalty \quad (25.37)$$

Il termine di tracking coppia R_torque incoraggia le dita a esercitare coppie prossime al target $\tau\_target = \tau\_max \cdot F\_cmd$, dove $F\_cmd \in [0,1]$ è il comando di forza di presa:

$$R\_torque = w\_\tau \cdot \Sigma\_\{i \in fingers\} \quad exp(-(\tau_i - \tau\_target)^2 / (2\sigma^2)) \cdot I\_contact\_i \quad (25.38)$$

dove I_contact_i è la funzione indicatore che vale 1 se il dito i è in contatto con l'oggetto. Per il pollice (biomeccanicamente asimmetrico), il reward è binario: $R\_torque\_thumb = I(\tau\_min \leq \tau\_thumb \leq \tau\_max)$. La forma gaussiana del reward per le altre dita penalizza continuamente le deviazioni dal target, con la larghezza σ che controlla la tolleranza.

Il termine R_consistency penalizza le differenze di flessione tra le quattro dita omologhe (indice, medio, anulare, mignolo):

$$R\_consistency = -w\_diff \cdot Var(\{q\_j \mid j \in inner\ fingers\}) - w\_outer \cdot \|q\_outer - c\_outer\|^2 \quad (25.39)$$

Il primo componente penalizza la varianza delle posizioni del primo giunto delle quattro dita (promuovendo grasping simmetrici); il secondo penalizza la deviazione dei giunti distali dal loro centro geometrico. Questi termini producono grasping a forma di cupola multi-punto che è sia esteticamente plausibile sia meccanicamente stabile — una proprietà essenziale per la trasferibilità sim-to-real, poiché i grasping a due dita irregolari tipicamente falliscono a causa di piccole differenze nelle proprietà di attrito della superficie.

25.15 Il Sim-to-Real come Problema di Transfer nel Latente

L'analisi delle sezioni precedenti ha mostrato che il sim-to-real gap si manifesta come un disallineamento tra le distribuzioni di transizione del simulatore e della realtà. Da un punto di vista di apprendimento delle rappresentazioni, questo disallineamento può essere visto come un caso specifico del problema del domain transfer nel latente: il modello apprende una mappa encoder $\varepsilon\_sim : X\_sim \to Z$ che produce rappresentazioni utili per la pianificazione nel dominio di simulazione, ma queste rappresentazioni potrebbero non allinearsi con quelle prodotte da $\varepsilon\_real : X\_real \to Z$ — il medesimo encoder applicato a osservazioni reali. Se le rappresentazioni latenti non sono allineate tra domini, le dinamiche latenti apprese $f\_\theta : Z \to Z$ in simulazione non saranno valide per la pianificazione nel dominio reale, anche se l'architettura del world model è corretta.

La soluzione più diretta è richiedere che l'encoder sia invariante al dominio: $\varepsilon\_sim(x\_sim) \approx \varepsilon\_real(x\_real)$ per coppie corrispondenti (x_sim,

x_real). Questo allineamento può essere enforced con un termine di perdita di consistenza dinamica nel latente (Yoneda et al. 2021):

$$\mathcal{L}_{align} = E_{(x_{sim}, a, x_{sim'})} [\| f_\theta(\varepsilon(x_{sim}), a) - stop\_grad(\varepsilon(x_{sim'})) \|^2] + \lambda \cdot E_{(x_{sim}, x_{real})} [\| \varepsilon(x_{sim}) - \varepsilon(x_{real}) \|^2] \quad (25.31)$$

Il primo termine è l'obiettivo di consistenza dinamica nel latente (equivalente all'obiettivo AC-JEPA della sezione 25.3): il predittore delle dinamiche f_θ deve essere coerente con le transizioni del simulatore. Il secondo termine allinea le rappresentazioni delle osservazioni simulate e reali corrispondenti (stessa configurazione del robot, stessa scena) — un obiettivo di contrastive/metric learning che richiede coppie (sim, real) corrispondenti, ottenibili da sessioni di calibrazione dove lo stesso scenario viene osservato sia nel simulatore che nella realtà.

Quando le coppie (sim, real) corrispondenti non sono disponibili, l'allineamento deve avvenire in modo non supervisionato attraverso tecniche di adversarial domain adaptation. Il discriminatore $D_{adv} : Z \to \{0,1\}$ è addestrato a distinguere tra rappresentazioni simulate e reali; l'encoder ε è addestrato a ingannare il discriminatore, producendo rappresentazioni indistinguibili per dominio:

$$\min_\varepsilon \max_{D_{adv}} E_{x_{sim}}[\log D_{adv}(\varepsilon(x_{sim}))] + E_{x_{real}}[\log(1 - D_{adv}(\varepsilon(x_{real})))] \quad (25.32)$$

La perdita avversariale (25.32), combinata con la perdita di consistenza dinamica (termine sinistro di 25.31), produce encoder che mappano osservazioni simulate e reali nello stesso sottospazio latente in modo che la dinamica latente appresa in simulazione sia valida anche nel dominio reale. Questo framework, noto come Latent Domain Adaptation for World Models, unifica la letteratura sul domain adaptation visivo con quella sui world model per la robotica: il sim-to-real gap diventa un problema di alignment delle distribuzioni nello spazio latente, risolvibile

con gli stessi strumenti algebrici e probabilistici usati nel transfer learning visivo (DANN, CORAL, MMD).

La connessione con l'architettura JEPA è immediata: poiché JEPA non ha un decoder, l'allineamento nel latente è il solo meccanismo disponibile per il trasferimento. I world model JEPA addestrati con l'obiettivo (25.31) producono rappresentazioni che catturano la struttura causale dell'ambiente — le invarianze rispetto a trasformazioni dello stile visivo che non cambiano le conseguenze delle azioni — in modo più robusto rispetto ai world model con decoder, che possono sprecare capacità rappresentazionale su dettagli visivi irrilevanti per il controllo.

26.1 Il Problema della Causalità nelle Rappresentazioni Apprese

Correlazione e intervento: la distinzione fondamentale

Le architetture descritte nelle parti precedenti di questa monografia — RSSM, JEPA, AC-JEPA — apprendono la propria rappresentazione del mondo minimizzando obiettivi predittivi su dati di esperienza. In tutti i casi, i dati provengono da una distribuzione osservazionale: l'agente osserva traiettorie (o_t, a_t, o_{t+1}) generate da una policy di comportamento (o di raccolta dati) fissa. Questo regime osservazionale è sufficiente per apprendere correlazioni statistiche tra stati, azioni ed esiti, ma non è sufficiente per apprendere la struttura causale del mondo — ovvero la struttura che risponde a domande del tipo: «cosa accadrebbe se eseguissi l'azione a, indipendentemente dalla policy che di solito produce a in questo contesto?»

Il linguaggio formale per rispondere a domande di questo tipo è il calcolo do di Pearl, introdotto nei tardi anni Novanta e sistematizzato nel libro «Causality» del 2000. Il calcolo do introduce l'operatore di intervento do(A = a) che descrive il valore della variabile A quando essa viene fissata

esternamente al valore a, rompendo le dipendenze causali provenienti dai genitori di A nel grafo causale. La distribuzione probabilistica indotta dall'intervento è in generale diversa dalla distribuzione osservazionale condizionata:

$$P(Y \mid do(A = a)) \neq P(Y \mid A = a) \quad (26.1)$$

La differenza tra i due termini è precisamente la confusione causale: la distribuzione condizionata $P(Y \mid A=a)$ include il contributo delle cause comuni di A e Y (i cosiddetti confounders), mentre la distribuzione interventistica $P(Y \mid do(A=a))$ le isola, fissando A esternamente e rompendo le frecce causali entranti. Per un agente robotico che apprende da dati osservazionali raccolti con una policy fissa, l'errore di confusione si traduce in una sottostima sistematica dell'effetto causale delle proprie azioni.

World Model causali e il do-calculus

Un world model causale è un modello che, oltre alla distribuzione di transizione $P(s_{t+1} \mid s_t, a_t)$ apprende dall'esperienza, è in grado di rispondere correttamente a query interventistiche del tipo $P(s_{t+k} \mid do(a_t = a), \ldots, do(a_{t+k-1} = a'))$. Formalmente, un Structural Causal Model (SCM) è una quadrupla (U, V, F, P_U), dove $U = \{U_1, \ldots, U_n\}$ sono le variabili esogene (rumore), $V = \{V_1, \ldots, V_m\}$ sono le variabili endogene (osservabili), $F = \{f_1, \ldots, f_m\}$ sono le funzioni strutturali $V_i = f_i(Pa(V_i), U_i)$ — dove $Pa(V_i)$ è l'insieme dei genitori causali di V_i nel grafo — e P_U è la distribuzione delle variabili esogene.

L'intervento $do(V_i = v)$ è modellato come la modifica del modello strutturale ottenuta sostituendo la funzione f_i con la funzione costante $V_i = v$, producendo il grafo mutilato in cui tutte le frecce causali entranti in V_i sono rimosse. La distribuzione interventistica $P(V \setminus V_i \mid do(V_i = v))$ si calcola nel modello mutilato. Nell'SCM, la regola di aggiustamento per backdoor consente di identificare la distribuzione interventistica da dati

osservazionali quando esiste un insieme di variabili W che soddisfa il backdoor criterion:

$$P(Y \mid do(X=x)) = \Sigma_w P(Y \mid X=x, W=w)\, P(W=w) \quad (26.2)$$

La formula (26.2) è il fondamento del deconfounding osservazionale: se si conosce l'insieme W di confounders (o un sottoinsieme sufficiente), è possibile stimare l'effetto causale senza mai eseguire esperimenti controllati. La sfida aperta nell'apprendimento automatico è che, in ambienti visivi complessi, le variabili W non sono note a priori e devono essere scoperte dalla struttura dei dati — un problema di causal discovery che rimane in larga misura irrisolto per rappresentazioni ad alta dimensionalità come le immagini.

Rappresentazione causale latente

Un approccio promettente al problema della causalità nelle rappresentazioni latenti è la Causal Representation Learning (CRL), che cerca di apprendere rappresentazioni latenti che corrispondano alle variabili causali del meccanismo generativo sottostante. Formalmente, si assume che i dati $x \in X$ siano generati da un insieme di variabili latenti causali $c = (c_1, \ldots, c_k)$ secondo un meccanismo generativo $x = g(c)$, e che le variabili c obbediscano a un SCM. L'obiettivo della CRL è apprendere un encoder $\varepsilon : X \to Z$ tale che $\varepsilon(x) \approx c$ — ovvero che le coordinate dello spazio latente corrispondano alle variabili causali del meccanismo generativo.

Kügelgen e colleghi (2021) e Schölkopf e colleghi (2021) hanno dimostrato condizioni di identificabilità della CRL: sotto certe ipotesi sulla struttura del grafo causale e sulla distribuzione delle perturbazioni, è possibile identificare le variabili causali latenti a meno di trasformazioni element-wise, ovvero separare le variabili causali ma non necessariamente ordinarle o scalarle. Questi risultati di identificabilità, pur non essendo ancora sufficientemente generali per applicazioni pratiche non supervisionate, forniscono garanzie teoriche che rendono la CRL una

direzione di ricerca promettente per la costruzione di world model intrinsecamente causali.

26.2 Composizionalità e il Problema del Binding

Composizionalità strutturale nelle rappresentazioni

La composizionalità è la proprietà per cui il significato di una struttura complessa è determinato dal significato delle sue parti e dalle regole della loro combinazione. Nei linguaggi naturali, la frase «il robot rosso porta la scatola blu» ha un significato che si ottiene componendo le rappresentazioni di «robot», «rosso», «porta», «scatola» e «blu» secondo la struttura sintattica della frase. Il sistema visivo umano possiede una capacità analoga: può riconoscere oggetti mai visti come combinazioni di proprietà note (colore, forma, materiale, posizione relativa) e può ragionare su scene composizionali che non ha mai osservato prima.

Le architetture neurali standard — reti convoluzionali, Transformer — rappresentano la scena visiva in modo distribuito: ogni feature della rappresentazione latente risponde a una combinazione non interpretabile di aspetti della scena, senza che esista una corrispondenza biunivoca tra posizioni del tensore latente e oggetti della scena. Questa modalità di rappresentazione distribuita è eccellente per il riconoscimento ma problematica per il ragionamento composizionale: per rispondere alla domanda «quanti oggetti rossi ci sono nella scena?» con un sistema a rappresentazione distribuita, la risposta deve essere emergente da pattern della rappresentazione globale piuttosto che essere computata direttamente da slot dedicati agli oggetti.

Slot Attention e rappresentazioni oggetto-centriche

Il meccanismo Slot Attention, introdotto da Locatello e colleghi (2020), propone un approccio oggetto-centrico alla rappresentazione: invece di codificare la scena in un singolo vettore latente, la scena viene

scomposta in K slot indipendenti, ciascuno dei quali compete per «catturare» le informazioni di uno specifico oggetto nella scena. Il meccanismo di attenzione iterato che distribuisce le informazioni della scena agli slot è definito come segue.

Sia $F \in \mathbb{R}^{N \times d}$feat il tensore delle feature estratte dall'encoder (N posizioni, d_{feat} canali) e $S \in \mathbb{R}^{K \times d}{}_{slot}$ la matrice degli slot. Ad ogni iterazione T dell'algoritmo di attenzione, gli slot vengono aggiornati come:

$$A_{kn} = softmax_k(q_k \cdot k_n / \sqrt{d}) \in \mathbb{R}^{K \times N} \quad (26.3)$$

$$A'_{kn} = A_{kn} / \Sigma_{n'} A_{kn'} \qquad (normalizzazione\ per\ slot) \quad (26.4)$$

$$updates_k = \Sigma_n A'_{kn} \cdot v_n \quad (26.5)$$

$$S_k \leftarrow GRU(S_k, updates_k) \quad (26.6)$$

dove $q_k = W^q S_k$, $k_n = W^k F_n$ e $v_n = W^v F_n$ sono le proiezioni delle feature e degli slot negli spazi di query, chiavi e valori. La normalizzazione (26.4), applicata lungo la dimensione degli slot anziché quella delle posizioni (come nel softmax standard), garantisce che gli slot competano per le feature piuttosto che le feature distribuirsi uniformemente tra gli slot. Questo meccanismo competitivo è la chiave della segmentazione emergente: slot diversi tendono a specializzarsi su regioni diverse della scena, producendo naturalmente una scomposizione oggetto-centrica senza supervisione.

Composizionalità e world model oggetto-centrici

L'integrazione della rappresentazione oggetto-centrica con i world model apre la strada ai cosiddetti Object-Centric World Model (OCWM), in cui le dinamiche sono modellate non al livello della scena globale ma al livello degli oggetti individuali. Nel framework OCWM, il modello di transizione latente è:

$$s^{(k)}_{t+1} = f_\theta(s^{(k)}_t, a_t, \{s^{(j)}_t\}_{j \neq k}) \quad k = 1, \ldots, K$$

(26.7)

dove $s^{(k)}_t$ è lo stato latente dell'oggetto k al tempo t e il modello di transizione può includere interazioni tra oggetti diversi (il termine $\{s^{(j)}_t\}$). Questa struttura è compositiva nel senso che il numero di parametri del modello cresce linearmente con il numero massimo di oggetti nella scena, anziché esponenzialmente come richiederebbe la modellazione dell'intera scena senza struttura oggetto-centrica. Quando un nuovo oggetto appare nella scena, un nuovo slot viene allocato e il suo slot latente viene inizializzato, senza dover riaddestrare l'intero modello.

Il limite principale degli OCWM attuali è la difficoltà di ottenere una segmentazione oggetto-centrica robusta in scene visive complesse, con illuminazione variabile, occlusioni parziali e texture simili tra oggetti diversi. I metodi attuali come SLATE e DINOSAUR (2023) hanno migliorato significativamente la qualità della segmentazione integrando feature semantiche estratte da encoder pre-addestrati (come DINO o CLIP) con il meccanismo Slot Attention, ma la robustezza nelle scene del mondo reale rimane un problema aperto.

26.3 Il Divario Simulazione-Realtà

Natura e tassonomia del sim-to-real gap

Il divario tra simulazione e realtà — noto nella letteratura anglosassone come sim-to-real gap — è il fenomeno per cui un sistema di controllo appreso in un ambiente simulato degrada le proprie prestazioni quando trasferito all'ambiente fisico reale. Questo degrado non è monolitico: origina da molteplici fonti di discrepanza tra simulatore e realtà che agiscono a scale diverse e che richiedono strategie di mitigazione complementari.

Le fonti principali di divario simulazione-realtà si articolano su quattro dimensioni. La prima è il divario visivo: i motori di rendering fotograficamente accurati (come NVIDIA Isaac Sim o Mujoco con ray tracing) non riproduzione fedelmente le proprietà ottiche degli oggetti reali, in particolare la variabilità delle texture, l'illuminazione complessa, i riflessi e la trasparenza. La seconda è il divario dinamico: i simulatori approssimano la fisica dei corpi rigidi con alta fedeltà, ma la fisica dei contatti, l'attrito anisotropo, la deformabilità dei materiali e le dinamiche dell'aria rimangono fonti significative di errore. La terza è il divario attuativo: i motori reali hanno saturazione, isteresi, elasticità delle trasmissioni, backlash e dinamiche ad alta frequenza che i modelli rigidi del simulatore ignorano. La quarta è il divario percettivo: i sensori reali — telecamere, LiDAR, encoders — sono soggetti a rumore, derive termiche, latenze e distorsioni non modelizzabili perfettamente.

Domain Randomization come stima del gap

La domain randomization, proposta da Tobin e colleghi nel 2017 e da Andrychowicz e colleghi nel 2020 nel contesto della manipolazione destrosa, affronta il sim-to-real gap addestrando la policy su una distribuzione di ambienti simulati $\varrho(\xi)$ parametrizzati da un vettore $\xi \in \Xi \subset \mathbb{R}^{d}_{\xi}$ che campiona diverse istanze dei parametri fisici e visuali. L'obiettivo di addestramento diventa:

$$max\_\pi \mathbb{E}\_\{\xi\sim\rho(\xi)\} \mathbb{E}\_\{\tau\sim\pi,Sim(\xi)\} [\Sigma\_t \gamma^t r\_t] \quad (26.8)$$

dove $Sim(\xi)$ denota il simulatore con parametri ξ. L'idea soggiacente è che se la distribuzione di simulatori $\varrho(\xi)$ è sufficientemente ampia da includere la realtà — ovvero se esiste $\xi^* \in \Xi$ tale che $Sim(\xi^*) \approx$ Realtà — allora una policy che funziona uniformemente bene per tutti gli $\xi \sim \varrho(\xi)$ sarà robusta anche nell'ambiente reale. Formalmente, questo corrisponde al principio di copertura: la realtà è vista come uno dei possibili mondi nel supporto di ϱ.

La scelta della distribuzione $\varrho(\xi)$ è critica e non banale. Una distribuzione troppo ristretta lascia la realtà fuori dal supporto (il gap non viene chiuso); una distribuzione troppo ampia rende il problema di ottimizzazione eccessivamente difficile, poiché la policy deve essere robusta a ambienti fisicamente irrealistici. Metodi di domain randomization adattiva (come ADR di OpenAI) regolano automaticamente i limiti di ξ in base alle prestazioni della policy, espandendo progressivamente la distribuzione man mano che la policy impara a gestire la variabilità corrente.

World Model per il trasferimento sim-to-real

I world model offrono una prospettiva diversa sul sim-to-real gap: invece di ignorare esplicitamente il divario (come nella domain randomization), il modello del mondo può essere usato per stimarne la struttura e correggerla. Nell'approccio di adattamento basato su world model, il sistema apprende un modello del simulatore nel latente durante il pre-addestramento, poi adatta i parametri del modello su un piccolo numero di interazioni reali. Formalmente, si definisce il bias del simulatore come:

$$\Delta_t = s^{real}_{t+1} - f_\theta(s_t, a_t) \quad (26.9)$$

dove s^{real}_{t+1} è lo stato osservato nell'ambiente reale e $f_\theta(s_t, a_t)$ è la predizione del modello addestrato in simulazione. Se il bias Δ_t è stazionario o lentamente variante nel tempo, può essere stimato con un residual model $r_\varphi(s_t, a_t) \approx \Delta_t$ addestrato su poche traiettorie reali. Il modello corretto $f_\theta + r_\varphi$ è poi usato per la pianificazione nel dominio reale. Questa strategia di Sim-to-Real via residual model è stata validata empiricamente in compiti di locomotione bipede (Hwangbo et al. 2019) e manipolazione (Qi et al. 2023), dove il residual model cattura la dinamica di contatto mancante nel simulatore con soli pochi minuti di dati reali.

Bayesian Uncertainty Quantification per il trasferimento robusto

Un framework alternativo e complementare alla domain randomization è la quantificazione esplicita dell'incertezza epistemica nel world model. Un modello del mondo bayesiano rappresenta la propria incertezza sui parametri θ tramite una distribuzione a posteriori $p(\theta|D)$ — dove D è il dataset di addestramento — e pianifica in modo robusto rispetto a questa incertezza. L'obiettivo di pianificazione robusto è:

$$a^* = \text{argmax}_{a_0,\ldots,a_{H-1}} \min_{\theta \sim p(\theta|D)} \mathbb{E}_\theta \left[\sum_t \gamma^t r_t \right] \tag{26.10}$$

Questo minimax corrisponde a una strategia di pianificazione pessimistica (worst-case) rispetto all'incertezza del modello — un'istanza del robusto controllo dell'incertezza che si connette alla teoria dei giochi a due giocatori: l'agente massimizza il ritorno mentre un «avversario» sceglie i parametri del modello per minimizzarlo. In pratica, l'ottimizzazione (26.10) è computazionalmente intrattabile esatta, ma viene approssimata tramite tecniche come il Bayesian Optimization, i Gaussian Process variazionali, o il Deep Ensemble (Lakshminarayanan et al. 2017), in cui $p(\theta|D)$ è approssimata con un insieme di modelli deterministici diversamente inizializzati.

26.3.1 Analisi Quantitativa del Divario Percettivo e Attuativo

Il divario percettivo nelle telecamere RGB: artefatti e rendering

Il divario percettivo tra simulatori fotograficamente avanzati (NVIDIA Isaac Sim con ray tracing, CARLA su Unreal Engine) e la realtà è stato quantificato da Aljalbout et al. (2026) usando la Fréchet Inception Distance (FID), che misura la distanza tra le distribuzioni di feature estratte da Inception-v3 su set di immagini simulate e reali. Su scene di cucina con robot manipolatore, FID(sim, real) ≈ 45–80 per Isaac Sim con texture fotorealistiche, versus FID ≈ 120–200 per MuJoCo con rendering OpenGL — confermando che l'investimento in rendering fotorealistico riduce ma non elimina il gap percettivo.

Gli artefatti specifici dei renderizzatori simulati che più contribuiscono al gap percettivo sono stati identificati attraverso ablation studies: l'assenza di aberrazione cromatica (che nelle lenti reali produce sfrangiamento colorato ai bordi) contribuisce per FID +8.3; l'assenza di rolling shutter (le telecamere CMOS reali acquisiscono righe dell'immagine in sequenza, non istantaneamente) per FID +12.7; la distorsione geometrica della lente (pin-cushion e barrel distortion) per FID +6.1; il motion blur su scene dinamiche per FID +15.4. Questi numeri suggerono che il motion blur è il fattore singolo più importante non modellato nei simulatori standard — non sorprendentemente, dato che le scene robotiche hanno movimenti rapidi del braccio e della telecamera montata sul gripper.

La soluzione pratica più efficace è la data augmentation con trasformazioni fotometriche durante l'addestramento: random color jitter, random grayscale, random blur e random cutout riducono il gap percettivo di FID +25–35 punti senza richiedere miglioramenti al renderizzatore. Questa strategia — combinata con il domain randomization visivo delle texture e dell'illuminazione — è la ricetta

standard per il controllo visuo-motore sim-to-real, documentata in IndustReal (Tang et al. 2023) e DexPoint (Qin et al. 2022).

Il divario attuativo: modelli di attuatore e latenza

Il divario attuativo è la componente del sim-to-real gap più spesso sottovalutata nelle fasi di progettazione e più spesso responsabile dei fallimenti di trasferimento in pratica. La survey di Aljalbout et al. (2026) identifica tre strati di complessità attuativa tipicamente assenti nei simulatori standard.

Il primo strato è la dinamica elettro-meccanica dell'attuatore: un motore DC reale non risponde istantaneamente al comando di coppia/corrente, ma ha una costante di tempo meccanica $\tau_{mech} = J_{rotor} / b$ (dove J_{rotor} è l'inerzia del rotore e b è il coefficiente di smorzamento viscoso) e una costante di tempo elettrica $\tau_{elec} = L/R$ (dove L e R sono induttanza e resistenza dell'avvolgimento). La risposta in frequenza del motore è un filtro del secondo ordine con due poli a $-1/\tau_{mech}$ e $-1/\tau_{elec}$, tipicamente nell'intervallo 10–100 Hz. I simulatori modellano il motore come sistema del primo ordine con $\tau = \tau_{mech}$, trascurando τ_{elec} — un'approssimazione valida a basse frequenze ma che introduce fase aggiuntiva non modellata ad alta frequenza, contribuendo all'instabilità nei cicli di controllo veloci.

Il secondo strato è la catena di trasmissione: riduttori armonici, viti a sfera, tendini in cavo, ruote dentate — tutti introducono non-linearità (backlash, isteresi, attrito di Stribeck con dipendenza dalla velocità) che i modelli rigidi del simulatore trascurano. Il modello di Stribeck per l'attrito dipendente dalla velocità è:

$$F_{friction}(v) = F_c \cdot sign(v) + (F_s - F_c) \cdot exp(-(v/v_s)^2) \cdot sign(v) + b \cdot v \quad (26.41)$$

dove F_c è la forza di attrito di Coulomb (costante con il segno della velocità), $F_s > F_c$ è la forza di attrito statico (maggiore alla velocità nulla

— il fenomeno dello stiction), v_s è la velocità di Stribeck (velocità alla quale l'attrito dinamico è minimo), e b è il coefficiente di smorzamento viscoso. Il fenomeno di stiction — il picco di attrito statico che deve essere superato per avviare il movimento — è la causa principale del jerk irregolare dei robot reali nelle inversioni di velocità, un comportamento che non si manifesta nei simulatori con modello di Coulomb isotropo.

Il terzo strato è la latenza del ciclo di controllo: nei sistemi reali, il tempo tra l'acquisizione del sensore e l'attuazione del comando è non trascurabile (tipicamente 5–30 ms per robot industriali, 1–5 ms per robot di ricerca con hardware dedicato) e variabile nel tempo (jitter della comunicazione, latenza del bus CAN, overhead del sistema operativo). I simulatori tipicamente assumono latenza zero o costante. Nei sistemi di controllo ad alta frequenza (500 Hz – 1 kHz per la locomozione bipede), una latenza non modellata di 5 ms equivale a 2.5–5 passi di controllo, che può destabilizzare completamente un controllore ottimizzato per latenza zero.

La soluzione proposta da Aljalbout et al. per il gap attuativo include tre componenti: (1) modellazione esplicita del delay inserendo una coda di azioni nella pipeline di simulazione e eseguendo l'azione t−d al posto dell'azione t, con d il delay stimato; (2) random noise sulla coppia dell'attuatore con distribuzione calibrata sull'hardware reale; (3) curriculum di addestramento che aumenta progressivamente il delay e il rumore man mano che la policy stabilizza.

26.4 Formalizzazione POMDP del Divario di Realtà: un Framework Unificato

Definizione formale del divario di realtà e del divario di prestazione

Il lavoro di Aljalbout e colleghi (Annual Review of Control, Robotics, and Autonomous Systems, 2026) propone la formalizzazione più

sistematica disponibile in letteratura del sim-to-real gap attraverso il framework dei Partially Observable Markov Decision Process (POMDP). Questa formalizzazione è preziosa perché distingue nettamente tra due concetti spesso confusi nella letteratura applicativa: il divario di realtà (reality gap), che è una proprietà intrinseca dei due ambienti indipendente dalla policy, e il divario di prestazione (performance gap), che dipende dalla specifica policy considerata.

Un POMDP è la tupla $M = (S, A, T, R, Z, O, \gamma)$, dove $S \subseteq \mathbb{R}^n$ è lo spazio degli stati, $A \subseteq \mathbb{R}^m$ è lo spazio delle azioni, $T : S \times A \to S$ è il modello di transizione, $R : S \times A \to \mathbb{R}$ è la funzione di ricompensa, $Z \subseteq \mathbb{R}^h$ è lo spazio delle osservazioni, $O : S \to Z$ è il modello di osservazione sensoriale, e $\gamma \in [0, 1)$ è il fattore di sconto. Si denota con M_s il POMDP del simulatore e con M_r il POMDP dell'ambiente reale.

Il divario di realtà è scomposto nelle sue due componenti fondamentali: il divario dinamico e il divario percettivo. Formalmente:

$$G_{dyn}(M_s, M_r) = E_{(s,a)\sim M_r} [D(T_{sim}(\cdot|s,a) \,\|\, T_{real}(\cdot|s,a))] \quad (26.11)$$

$$G_{perc}(M_s, M_r) = E_{s\sim S_r} [D(O_{sim}(\cdot|s) \,\|\, O_{real}(\cdot|s))] \quad (26.12)$$

dove D è una misura di divergenza tra distribuzioni di probabilità (tipicamente la divergenza di Kullback-Leibler o la variazione totale) e la notazione $(s, a) \sim M_r$ indica che stati e azioni sono campionati dalle distribuzioni marginali dell'ambiente reale. Il divario di prestazione per una specifica policy π è:

$$G_{perf}(M_s, M_r, \pi) = |J_{M_s}(\pi) - J_{M_r}(\pi)| \quad (26.13)$$

dove $J_M(\pi) = E_{\tau\sim\pi,M}[\Sigma_t \gamma^t R(s_t, \pi(b_t))]$ è il ritorno atteso di π nell'ambiente M, con b_t la belief sull'insieme degli stati. L'obiettivo del trasferimento sim-to-real è minimizzare il divario di prestazione:

$$\pi^* = \operatorname{argmin}_\pi \; G_{perf}(M_s, M_r, \pi) \quad (26.14)$$

Teorema di bound sul divario di prestazione

Un risultato fondamentale che connette divario di realtà e divario di prestazione è il seguente bound, derivabile dall'analisi di sensibilità dei processi di Markov (Kakade e Langford, 2002; Eysenbach et al., 2021). Per politiche deterministiche con orizzonte T e ricompense bounded in [0, R_max]:

$$G\_perf(M\_s, M\_r, \pi) \leq 2R\_max \cdot T \cdot [G\_dyn + G\_perc] + O(T^2 \cdot G^2\_dyn) \quad (26.15)$$

La dimostrazione procede per induzione sull'orizzonte T. Al passo base T = 1, la differenza di valore è bounded dalla differenza di transizione per il simulation lemma. Al passo induttivo, si usa il fatto che la distribuzione sui percorsi diverge esponenzialmente con T in presenza di G_dyn > 0. Il termine $O(T^2 \cdot G^2\_dyn)$ riflette gli errori composti del modello: a ogni passo, l'errore di modellizzazione si accumula, producendo distribuzioni sui percorsi che divergono quadraticamente con l'orizzonte.

Questa stima ha implicazioni pratiche importanti: per task a lungo orizzonte (T largo), anche un divario di realtà moderato G_dyn può produrre un divario di prestazione catastrofico. Questo spiega perché la locomotione bipede e la pianificazione a lungo termine sono particolarmente sensibili al sim-to-real gap: l'accumulo degli errori dinamici lungo la traiettoria porta il sistema reale in regioni dello spazio degli stati mai visitate in simulazione.

Una corollaria importante è che il divario di prestazione può essere zero anche quando il divario di realtà non è zero, purché la policy π sia sufficientemente robusta: se π non sfrutta le regioni dello spazio degli stati dove M_s e M_r divergono, G_perf = 0 indipendentemente da G_dyn. Questa osservazione giustifica l'approccio della domain randomization,

che addestra policy robuste a una famiglia di dinamiche invece di approssimare esattamente la realtà.

Classificazione sistematica delle fonti del divario

La survey di Aljalbout et al. (2026) classifica le fonti del divario di realtà in quattro categorie ortogonali, corrispondenti ai quattro moduli del pipeline robotico:

Il divario dinamico G_dyn nasce dalle approssimazioni del simulatore fisico: integrazione numerica con passo temporale finito, modellazione semplificata dei contatti corpo-corpo (spesso convex-hull anziché mesh completa), trascuramento della deformabilità dei materiali, approssimazione dell'attrito con il modello Coulomb isotropo, e ignoranza delle forze aerodinamiche. Per la robotica di contatto, il modello di contatto è la principale fonte di G_dyn: le simulazioni tipiche usano motori fisici basati su metodi di tipo LCP (Linear Complementarity Problem) che assumono corpi rigidi perfetti, mentre i materiali reali hanno deformabilità viscoelastica che modifica significativamente la forza e la direzione dei contatti.

Il divario percettivo G_perc origina dall'imperfezione della simulazione sensoriale: le telecamere reali hanno aberrazioni ottiche, rumore shot, motion blur e degradazione cromatica con la temperatura che nessun rendering corrente riproduce fedelmente; i LiDAR reali hanno ghosting su superfici riflettenti e perdita di ritorni su superfici troppo assorbenti. Il divario attuativo, strettamente connesso a G_dyn, riguarda la differenza tra il modello del motore nel simulatore e il motore reale: saturazione, elasticità delle trasmissioni, isteresi, deadzone e dinamiche termo-dipendenti dell'attività dell'avvolgimento sono tipicamente ignorati. Il divario di sistema, infine, copre differenze strutturali: il simulatore assume che i parametri del modello (masse, inezie, lunghezze

dei link) siano noti esattamente, mentre i valori reali sono affetti da incertezze costruttive di tipicamente ±5–10%.

26.5 Framework Bi-Livello per l'Ottimalità Sim-to-Real

Il problema dell'objective mismatch e la soluzione bi-livello

Anand e colleghi (2025) hanno identificato un problema fondamentale che affligge l'intera categoria dei metodi di domain randomization e domain adaptation: questi metodi ottimizzano proxy per il divario di realtà (accuratezza del simulatore, variabilità dei parametri, fedeltà predittiva) che non necessariamente correlano con la prestazione reale della policy. Formalmente, il problema dell'objective mismatch consiste nel fatto che il criterio di ottimizzazione del simulatore min_{θ_sim} E[D(T_sim(·|s,a;θ_sim), T_real(·|s,a))] è ortogonale all'obiettivo reale min_{θ_sim} G_perf(M_s(θ_sim), M_r, π*_{θ_sim}), dove π*_{θ_sim} è la policy ottimale nel simulatore con parametri θ_sim.

Per costruire un simulatore ottimale nel senso del trasferimento, non basta che il simulatore sia accurato nella norma L2 sulle transizioni: basta che il simulatore sia tale che la policy ottimale in esso sia anche ottimale nel mondo reale. Questo concetto è catturato dal principio di equivalenza di valore (Value Equivalence Principle di Grimm et al. 2020): due modelli che inducono la stessa funzione di valore ottimale sono equivalenti per gli scopi del controllo, indipendentemente dalla loro accuratezza sulle transizioni.

La proposta di Anand et al. è di riformulare il problema dell'adattamento del simulatore come un problema di reinforcement learning bi-livello:

$$\max_{\theta_{sim},\, \rho_r} J_{M_r}(\pi^*_{\theta_{sim}}) \quad \text{s.t.} \quad \pi^*_{\theta_{sim}} = \arg\max_{\pi} J_{M_s(\theta_{sim})}(\pi) \tag{26.16}$$

dove θ_sim sono i parametri del simulatore, ϱ_r è la ricompensa nel simulatore (anch'essa adattabile), e il vincolo esprime che la policy dell'inner level è ottimale nel simulatore. Il livello esterno (outer-level) usa dati reali per adattare θ_sim e ϱ_r in modo da massimizzare le prestazioni reali della policy indotta dall'inner level.

Derivazione del gradiente bi-livello tramite differenziazione implicita

La sfida computazionale del framework bi-livello è che il gradiente dell'obiettivo outer rispetto a θ_sim richiede di differenziare attraverso il processo di ottimizzazione dell'inner level. Formalmente, si considera il caso in cui la policy inner è appresa con Stochastic Policy Gradient (SPG):

$$\nabla_\varphi J_{M_s}(\pi_\varphi) = E_{\tau\sim\pi_\varphi, M_s} [\Sigma_t \nabla_\varphi \log \pi_\varphi(a_t|s_t) \cdot G_t] \quad (26.17)$$

Al convergenza dell'inner level, i parametri φ* soddisfano la condizione di punto fisso $\nabla_\varphi J_{M_s}(\pi_{\varphi^*}) = 0$. Il gradiente implicito dell'outer level rispetto a θ_sim è:

$$dJ_r/d\theta_{sim} = \partial J_r/\partial\varphi * (d\varphi^*/d\theta_{sim}) = -\partial J_r/\partial\varphi \cdot (\partial^2 J_s/\partial\varphi^2)^{-1} \cdot \partial^2 J_s/\partial\varphi\partial\theta_{sim} \quad (26.18)$$

dove la seconda uguaglianza segue dall'Implicit Function Theorem: se la matrice Hessiana $\partial^2 J_s/\partial\varphi^2$ è non singolare (condizione di regolarità soddisfatta in un intorno del massimo), allora φ* è una funzione differenziabile di θ_sim e la sua derivata è data dall'inversa dell'Hessiano del problema interno moltiplicata per la derivata mista. Il termine $(\partial^2 J_s/\partial\varphi^2)^{-1}$ è computazionalmente costoso (richiede l'inversione di una matrice $n_\varphi \times n_\varphi$, dove n_φ è il numero di parametri della policy), ma può essere approssimato con metodi di Neumann series o con il metodo di coniugato gradiente iterativo.

La sensibilità $d\varphi^*/d\theta_{sim}$ ha un'interpretazione intuitiva importante: misura quanto cambia la policy ottimale del simulatore quando si perturbano i parametri del simulatore. Se questa sensibilità è

alta, piccole modifiche a θ_{sim} producono policy molto diverse, il che significa che il simulatore è critico per il trasferimento. Se è bassa, la policy è robusta alle variazioni di θ_{sim} — un indicatore di buona generalizzazione.

Gli autori dimostrano che, sotto condizioni di regolarità, l'algoritmo bi-livello converge a una coppia (θ^*_{sim}, π^*) che soddisfa la condizione di ottimalità sim-to-real: $J_{M_r}(\pi^*) \geq J_{M_r}(\pi_{DR}) - \varepsilon$, dove π_{DR} è la policy di domain randomization e $\varepsilon \to 0$ con un numero sufficiente di interazioni nell'ambiente reale. Questa garanzia non è raggiungibile dai metodi di domain randomization standard, che possono restare permanentemente subottimali se la distribuzione di randomizzazione non include la realtà nel suo supporto.

26.6 Il Gemello Digitale Dinamico: il Paradigma Real-is-Sim

Architettura di sincronizzazione in tempo reale

Il framework Real-is-Sim, proposto da Abou-Chakra e colleghi (2025), inverte il paradigma convenzionale del sim-to-real: invece di addestrare una policy in un simulatore fisso e trasferirla alla realtà, mantiene un simulatore dinamico che si sincronizza in tempo reale con il mondo fisico, delegando a questo gemello digitale l'intera responsabilità del trasferimento. La policy interagisce sempre e soltanto con il simulatore — anche durante il dispiegamento in ambienti reali.

Il componente tecnologico centrale è Embodied Gaussians, il primo simulatore a realizzare una sincronizzazione in tempo reale con ambienti fisici in modo generalizzabile. Il simulatore combina due tecnologie: Gaussian Splatting per la ricostruzione e rendering fotorealistico della scena, e simulazione fisica basata su particelle per le dinamiche degli oggetti. Il gemello digitale si aggiorna a 60 Hz utilizzando osservazioni

provenienti dalle telecamere reali, che correggono continuamente le posizioni degli oggetti nel simulatore.

Il flusso di informazioni nel sistema Real-is-Sim è il seguente: la policy π_s riceve osservazioni derivate dallo stato del simulatore (pose degli oggetti, immagini renderizzate da telecamere virtuali), produce un'azione sulla rappresentazione simulata del robot, e il robot reale insegue i joint states del robot simulato attraverso un controllore di basso livello. Le osservazioni reali aggiornano il simulatore, mantenendo il suo stato allineato con la realtà. Formalmente, il problema di tracking del gemello digitale è:

$$\theta\_sim(t+1) = \theta\_sim(t) + \eta \cdot \nabla\_\{\theta\_sim\} \| O\_real(s\_\{t+1\}) - O\_sim(s\_\{t+1\}; \theta\_sim(t)) \|^2 \quad (26.19)$$

dove θ_sim sono i parametri del simulatore (pose degli oggetti, proprietà fisiche), O_real sono le osservazioni sensoriali reali e O_sim sono le osservazioni simulate. Il gradiente è calcolato tramite differenziazione del rendering (differentiable rendering) per la componente visuale e differenziazione dell'integratore fisico per la componente dinamica.

La conseguenza più importante di questo design è che il sim-to-real gap viene completamente trasferito dal problema della policy al problema della sincronizzazione: la policy non deve mai essere robusta al divario tra simulazione e realtà, perché il simulatore è sempre sincronizzato con la realtà. Questo permette di addestrare policy con rappresentazioni dello stato ricche (pose precisi degli oggetti, configurazione esatta dello spazio di lavoro) senza dover preoccuparsi della loro invarianza alla variabilità visiva dell'ambiente reale.

Confronto con comportamento cloning standard

Gli esperimenti di Abou-Chakra et al. su un task di manipolazione a lungo orizzonte (PushT) dimostrano che la politica Real-is-Sim trained

su rollout simulati raggiunge prestazioni comparabili alla politica addestrata su dati reali, con il vantaggio di poter generare dati sintetici in parallelo a costo quasi nullo. Il framework consente inoltre una valutazione offline affidabile delle politiche prima del dispiegamento reale: poiché il simulatore è sincronizzato con la realtà, le valutazioni nel simulatore sono predittive delle prestazioni reali — un requisito critico per la validazione di sistemi robotici in ambienti industriali.

Una limitazione attuale del paradigma è la sua dipendenza dalla qualità della sincronizzazione: se il gemello digitale si desincronizza dalla realtà (a causa di occlusioni, oggetti non modellati, o dinamiche impreviste), la policy riceve osservazioni che non corrispondono più alla realtà e fallisce. La robustezza della sincronizzazione è quindi il collo di bottiglia del sistema — un problema di percezione e stima dello stato, non di apprendimento per rinforzo.

26.7 Identificazione di Sistema Differenziabile per il Trasferimento sotto Carichi

Il problema del payload come divario strutturato

Wang e colleghi (2026) affrontano uno scenario di sim-to-real gap particolarmente insidioso per la robotica industriale: i robot umanoidi che operano in ambienti reali devono spesso trasportare carichi sconosciuti, che alterano sistematicamente la massa totale, il baricentro e il tensore di inerzia del sistema. Questo è un esempio di sim-to-real gap strutturato: il mismatch non è casuale (come il rumore attuativo) ma è sistematico e fisicamente interpretabile. Un framework di domain randomization standard tratterebbe questo mismatch come un disturbo da coprire con una distribuzione ampia, producendo policy conservative — il modello HALO (HeAvy-LOaded humanoid motion control) propone invece di identificare il payload prima dell'addestramento della policy, usando la simulazione differenziabile.

Identificazione a due stadi con MuJoCo XLA

Il framework HALO usa MuJoCo XLA (differentiable physics), una versione di MuJoCo che permette di calcolare gradienti analitici rispetto ai parametri fisici attraverso l'intero orizzonte di simulazione. Il problema di identificazione è formulato come minimizzazione della discrepanza tra traiettorie simulate e traiettorie reali:

$$\theta^* = \arg\min_{\theta} \Sigma_{t=1}^{T} \| q_t^{real} - q_t^{sim}(\theta) \|^2_{W} + \lambda_{reg} \|\theta - \theta_0\|^2 \quad (26.20)$$

dove q_t^{real} sono le configurazioni articolari reali, q_t^{sim}(θ) sono le configurazioni simulate con parametri θ, W è una matrice di pesi (che penalizza maggiormente le articolazioni portanti), θ_0 sono i valori nominali del modello CAD, e λ_reg è un termine di regolarizzazione che previene soluzioni fisicamente inconsistenti. Il gradiente ∂q_t^{sim}/∂θ è calcolato automaticamente dalla differenziazione del simulatore attraverso l'intero rollout di lunghezza T.

Lo schema a due stadi è:

Stadio 1 — Calibrazione del modello nominale: senza payload, si raccolgono traiettorie sotto una policy di esplorazione a piede fisso (single-foot constraint, che massimizza l'escursione articolare). Il modello nominale θ_base viene identificato minimizzando (26.20). La calibrazione rimuove il mismatch intrinseco dovuto alle tolleranze costruttive e alle proprietà dei materiali.

Stadio 2 — Identificazione del payload: con il modello calibrato θ_base come prior, si raccolgono traiettorie con payload e si identifica θ_payload. Poiché il modello base è già calibrato, lo stadio 2 ottimizza soltanto i parametri del payload (massa, posizione del baricentro rispetto al link di attacco, tensore di inerzia) — un problema a 10 gradi di libertà invece degli oltre 200 del robot completo.

Il gradiente del costo (26.20) rispetto ai parametri fisici richiede la differenziazione delle equazioni del moto di Newton-Euler attraverso l'integratore. Per un sistema ad N link con equazioni del moto $M(\theta)\ddot{q} + C(q, \dot{q}, \theta)\dot{q} + G(q, \theta) = \tau$, il gradiente rispetto alla massa m_i del link i è:

$$\partial \ddot{q}_t / \partial m_i = -M(\theta)^{-1} \cdot (\partial M / \partial m_i \cdot \ddot{q}_t + \partial C / \partial m_i \cdot \dot{q}_t + \partial G / \partial m_i) \quad (26.21)$$

dove le derivate $\partial M/\partial m_i$, $\partial C/\partial m_i$, $\partial G/\partial m_i$ sono le Jacobiane delle matrici di inerzia, Coriolis e gravità rispetto alla massa del link i. Queste derivate sono sparse (ciascuna massa influenza solo i termini che coinvolgono il link i e i suoi discendenti nella catena cinematica) e possono essere calcolate efficientemente con algoritmi ricorsivi analoghi all'algoritmo di Newton-Euler al contrario (recursive Newton-Euler in reverse, RNER).

I risultati di HALO dimostrano che il trasferimento zero-shot di policy addestrate con il modello identificato (invece che con domain randomization) riduce l'errore di tracking della traiettoria di un fattore 3–5x in condizioni di carico elevato (fino a 5 kg per un robot umanoide da 20 kg), e permette di eseguire compiti agili (walking rapido, cambio di direzione) che falliscono completamente con policy addestrate con domain randomization sullo stesso range di carichi.

26.8 Regolarizzazione Guidata dalla Fisica e Domain Randomization Condizionata

Il problema dell'agente di fortuna nella DR classica

Kawachi (2025) articola una critica sistematica alla domain randomization classica attraverso un'analisi statistica: per task che richiedono un divario di realtà ampio (gearbox ad alto rapporto di riduzione, attuatori con isteresi pronunciata, dinamiche di contatto aggressive), la probabilità che un campione $\xi \sim \varrho(\xi)$ dalla distribuzione di

randomizzazione coincida esattamente con i parametri reali ξ* è bassa, e la policy deve coprire una regione dell'insieme Ξ che include molti punti fisicamente irrealistici. Questo porta a due patologie osservate sperimentalmente: oscillazioni persistenti del controllore (il controllore impara a essere instabile in alcune regioni di Ξ per evitare la penalità in altre) e conservatorismo eccessivo (il controllore impara comportamenti a bassa ampiezza per essere sicuro nell'intera distribuzione).

Gain regularization come ancora fisica

La soluzione proposta è di re-interpretare i guadagni proporzionali del controllore PID classico (k_x, k_v, k_θ, k_ω) come surrogati misurabili delle dinamiche non modellate dell'hardware: invece di tentare di modellare esplicitamente l'isteresi, l'elasticità della trasmissione e il ritardo attuativo, questi effetti si manifestano come un guadagno proporzionale effettivo ridotto rispetto a quello teorico del modello rigido. Si misura il guadagno effettivo con esperimenti semplici di risposta a gradino sull'hardware reale:

$$k\_x^{eff} = lim\_{\varepsilon \to 0}\ \Delta f\_x / \Delta x \quad , \quad k\_\theta^{eff} = lim\_{\varepsilon \to 0}\ \Delta \tau / \Delta \theta \tag{26.22}$$

La funzione di perdita di gain regularization penalizza la deviazione delle sensibilità input-output del controllore neurale dai guadagni effettivi misurati:

$$\mathcal{L}\_reg(\pi\_\varphi) = \Sigma\_{d \in \{x,v,\theta,\omega\}}\ \left(\partial \pi\_\varphi(s)/\partial s\_d \ - k\_d^{eff} \right)^2 \cdot w\_d \tag{26.23}$$

dove s_d è la componente d dello stato (posizione, velocità, angolo, velocità angolare) e w_d è un peso che bilancia l'importanza relativa di ciascun grado di libertà. La sensibilità ∂π_φ/∂s_d è calcolata come Jacobiana della policy rispetto all'input, valutata nell'intorno del punto di equilibrio. L'obiettivo totale di addestramento è:

$$\min\_\varphi\ E\_{\xi \sim \rho(\xi)}[\ \mathcal{L}\_RL(\pi\_\varphi; \xi)\ ] \ + \ \beta \cdot \mathcal{L}\_reg(\pi\_\varphi) \tag{26.24}$$

dove $\mathcal{L}$_RL è la perdita di reinforcement learning standard e β è un coefficiente di bilanciamento.

Il complemento alla gain regularization è la domain randomization condizionata ai parametri: il parametro ξ del simulatore viene aggiunto all'input della policy, $\pi_\varphi(a|s, \xi)$, permettendo alla rete neurale di specializzare il proprio comportamento a ogni istanza dell'ambiente invece di produrre un unico compromesso. Questa condizionatura trasforma il problema da «apprendere una policy robusta per tutta la distribuzione» a «apprendere un mapping da (s, ξ) ad azioni ottimali per ciascun (s, ξ)» — un problema significativamente più semplice.

L'esperimento sul robot birotante con rapporto di riduzione 110:1 dimostra che la gain regularization riduce l'errore di assestamento angolare dell'87% rispetto alla DR standard, eliminando le oscillazioni persistenti che affliggevano la DR pura. La condizionatura ai parametri riduce ulteriormente il conservatorismo, portando il sistema a prestazioni del 95% di quelle raggiungibili con un controllore PID classico tarato manualmente sull'hardware — ma con la flessibilità non lineare di una rete neurale ricorrente.

26.8.1 Modello Cart-Pole con Attrito e Deriva della Gain Regularization

Il sistema cart-pole ad alto rapporto di riduzione come banco di prova

Il sistema birotante bilanciante (two-wheeled balancing robot) con rapporto di riduzione 110:1 analizzato da Kawachi (2025) è un banco di prova particolarmente severo per la gain regularization perché combina tre fattori sfavorevoli: il rapporto di riduzione elevato introduce forte isteresi meccanica e attrito di Stribeck, la dinamica di cart-pole invertito è intrinsecamente instabile (polo instabile in $s = +\sqrt{(g/l)}$ per il modello

linearizzato), e il sistema richiede cicli di controllo rapidi (10–30 ms) dove la latenza non modellata è critica.

Il modello non lineare del cart-pole con attrito viscoso è descritto dalle equazioni (19)-(23) del paper di Kawachi, che equivalgono alle equazioni del moto di Newton per la massa del carrello (massa M = m_cart + m_pole) e del pendolo (massa m_pole, lunghezza l_pole):

$$\Delta = M(m\,l^2 + J) - (ml\cos\theta)^2 \quad , \quad \textit{dove } \Delta \textit{ è il determinante inerziale dell'accoppiamento} \tag{26.42}$$

$$\Delta\,\ddot{x} = (ml)^2 g \sin\theta\cos\theta + (ml^2 + J)(f + ml\,\theta^2\sin\theta - D_c\,\dot{x}) + ml\,D_p\,\theta\cos\theta \tag{26.43}$$

$$\Delta\,\ddot{\theta} = Mmlg\sin\theta - ml\cos\theta\,(f + ml\,\theta^2\sin\theta - D_c\,\dot{x}) - M\,D_p\,\theta \tag{26.44}$$

dove x e θ sono rispettivamente la posizione orizzontale e l'angolo del pendolo, f è la forza applicata alle ruote, J è il momento di inerzia del pendolo attorno al suo baricentro, e D_c, D_p sono i coefficienti di smorzamento viscoso del carrello e del pendolo. Il modello (26.42)–(26.44) è accoppiato ($\ddot{x}$ e $\ddot{\theta}$ dipendono entrambi da f) e non lineare (termini sin θ, cos θ, θ^2).

La gain regularization di Kawachi (eq. 26.23 della monografia) misura i guadagni effettivi dell'hardware con esperimenti di risposta a gradino sull'angolo θ e sulla posizione x, ottenendo empiricamente: $k_x \in [1, 20]$ N/m, $k_v \in [12, 17]$ N/(m/s), $k_\theta \in [25, 40]$ N/rad, $k_\omega \in [1.3, 3.0]$ N/(rad/s). Questi valori riflettono l'intero stack di non-linearità del sistema fisico: il modello rigido nominale suggerirebbe guadagni proporzionali più elevati, ma l'attrito Stribeck, il backlash del riduttore e la latenza effettiva del ciclo li riducono. Penalizzare la deviazione delle sensibilità del controllore neurale da questi valori misurati ha quindi l'effetto di impedire al RL di apprendere comportamenti che sfruttano guadagni irrealisticamente elevati validi solo in simulazione.

Il risultato sperimentale chiave è che il controllore RNN con gain regularization ha un errore di assestamento angolare di 0.8° (RMS) nell'hardware reale, contro 7.2° del controllore RNN senza regularization e 5.4° del controllore PID classico — dimostrando che la gain regularization produce un controllore che è sia più preciso del PID (grazie alla componente non lineare della RNN) sia robusto al trasferimento (grazie all'ancoraggio fisico dei guadagni). La domain randomization condizionata ai parametri (eq. 26.24) aggiunge un ulteriore 35% di riduzione dell'errore di assestamento per configurazioni del pendolo con massa e lunghezza diverse da quelle nominali.

26.9 Imitazione dell'MPC con Robust Tube e Input Refinement Governor

Il compromesso computazionale del Model Predictive Control

Il Model Predictive Control (MPC) rappresenta il gold standard del controllo con vincoli: a ogni istante, risolve un problema di ottimizzazione su un orizzonte ricedente per calcolare la sequenza di azioni ottimale soggetta a vincoli sullo stato e sull'ingresso. La sua applicabilità in robotica real-time è limitata dalla complessità computazionale $O(n^3)$ del QP sottostante, che per sistemi con decine di gradi di libertà richiede tipicamente 10–100 ms per soluzione — un ordine di grandezza superiore ai cicli di controllo richiesti dalla robotica ad alta velocità (1–5 ms).

Kim e colleghi (2026) propongono di approssimare l'MPC con una rete neurale profonda addestrata per imitazione (imitation learning), riducendo il tempo di inferenza di un fattore 10–20x. Questa imitazione introduce inevitabilmente il sim-to-real gap: i dati di addestramento per l'imitazione sono raccolti in simulazione (dove è pratico eseguire migliaia di soluzioni MPC), ma la DNN viene poi deployata nell'ambiente reale. Il contributo del lavoro è un framework di controllo che gestisce questo gap senza richiedere ulteriori dati reali.

Robust Tube MPC: la teoria del tubo di raggiungibilità

Il Robust Tube MPC (RTMPC) di Mayne e colleghi (2005) è un approccio di controllo robusto che garantisce la soddisfazione dei vincoli in presenza di perturbazioni bounded. L'idea centrale è di separare il problema nominale (senza perturbazioni) dal problema robusto (con perturbazioni), usando un «tubo» di traiettorie raggiungibili centrato sulla traiettoria nominale.

Sia x il vettore di stato e u l'ingresso di controllo. Le dinamiche perturbate sono:

$$x_{t+1} = f(x_t, u_t) + w_t \quad , \quad w_t \in W \quad \textit{(perturbazione bounded)} \tag{26.25}$$

L'MPC nominale risolve il problema con w_t = 0 e produce la traiettoria nominale {x̄_t, ū_t}. Il controllore di feedback K(x_t − x̄_t) gestisce le deviazioni dalla traiettoria nominale dovute alle perturbazioni. L'ingresso totale applicato è:

$$u_t = \bar{u}_t + K \cdot (x_t - \bar{x}_t) \quad , \quad con \quad \|x_t - \bar{x}_t\| \leq \delta \quad \forall t \tag{26.26}$$

dove δ è il raggio del tubo di raggiungibilità: l'insieme delle deviazioni dalla traiettoria nominale che il feedback K è in grado di compensare, garantendo che il vincolo sullo stato sia soddisfatto da tutti gli stati nel tubo. Nel framework di Kim et al., il sim-to-real gap è modellato come una perturbazione w_t = f_real(x_t, u_t) − f_sim(x_t, u_t) bounded da un insieme W stimato offline.

L'Input Refinement Governor

Un limite dell'imitazione di MPC con reti neurali è che la DNN non garantisce la soddisfazione dei vincoli: anche se è addestrata su traiettorie MPC che li rispettano, una DNN può produrre azioni che violano i vincoli in stati non visti durante l'addestramento (distributional shift). L'Input Refinement Governor (IRG) proposto dagli autori è un modulo

supervisore che riceve l'azione proposta dalla DNN e la proietta sull'insieme delle azioni ammissibili vincolate:

$$u_t^{refined} = \arg\min_{u \in U_t} \|u - u_t^{DNN}\|^2 \quad \text{s.t.} \quad U_t \subseteq \{ u : x_{t+1}(u) \in X_t \} \quad (26.27)$$

dove U_t è l'insieme delle azioni ammissibili al tempo t (determinato dalla soluzione del Robust Tube MPC nominale e dal raggio del tubo δ), e x_{t+1}(u) è la predizione a un passo del modello nominale. La proiezione (26.27) è un QP a bassa dimensione (dimensione dell'ingresso u) che richiede tipicamente < 0.1 ms e quindi non compromette la velocità del ciclo di controllo.

L'effetto dell'IRG è di mantenere la DNN nell'intorno della traiettoria nominale (il tubo), prevenendo lo drift fuori distribuzione che porta ai fallimenti di controllo. Gli esperimenti su cart-pole e controllo di un veicolo autonomo in situazioni di evitamento della collisione mostrano che il framework RTMPC+DNN+IRG richiede in media il 2% del tempo computazionale dell'MPC puro mantenendo il 98% delle prestazioni e garantendo la soddisfazione dei vincoli con probabilità > 0.99.

26.10 Manipolazione Destrosa Tattile: Chiusura Zero-Shot del Gap

Il caso della manipolazione in-hand come problema aperto

Mentre il sim-to-real transfer per la locomozione ha raggiunto un livello di maturità che ne consente l'uso industriale (ANYmal, Figure, Unitree), la manipolazione destrosa rimane un problema sostanzialmente aperto. La differenza fondamentale è di natura fisica: la locomozione coinvolge contatti corpo-suolo che sono relativamente ben modellabili con la dinamica dei corpi rigidi (le forze di contatto sono principalmente normali con attrito di Coulomb), mentre la manipolazione in-hand coinvolge multi-point contacts tra dita e oggetti, con forze di attrito

anisotrope, deformabilità dei cuscinetti delle dita e delle superfici degli oggetti, e variabilità estrema delle proprietà dei materiali.

Il lavoro di ByteDance Seed (2025) presenta il primo sistema che realizza una manipolazione destrosa con zero-shot sim-to-real transfer su una mano a cinque dita, senza alcun fine-tuning su dati reali. Le due capacità dimostrate — grasp force tracking controllabile e reorientamento in-hand di oggetti — richiedono entrambe una modellazione precisa della coppia dinamica percezione-attuazione.

Simulazione tattile ad alta frequenza

Il componente critico del sistema è la simulazione tattile: la mano destrosa ha 64 unità tattili per dito (totale 320 per la mano completa), ciascuna delle quali misura la forza normale di contatto locale. Riprodurre queste misure in simulazione con sufficiente fedeltà da permettere l'addestramento di policy tattili è il problema principale.

La soluzione degli autori evita la simulazione fisica di ogni unità tattile (computazionalmente intractable) e usa invece un modello geometrico: la forza tattile sull'unità k è approssimata come funzione della distanza d_k tra il punto di contatto virtuale k sul dito e la mesh dell'oggetto, calcolata tramite forward kinematics parallela:

$$f_k^{tactile} = max(0, -\kappa \cdot d_k) \quad , \quad d_k = distance(p_k^{finger}(q), surface_{obj}) \quad (26.28)$$

dove κ è la rigidezza di contatto, q è la configurazione articolare della mano, e p_k^{finger}(q) è la posizione del k-esimo punto tattile ottenuta tramite cinematica diretta. Il calcolo delle 320 distanze è parallelizzato su GPU, raggiungendo una frequenza di aggiornamento di 2.000 Hz — sufficiente per il ciclo di controllo tattile della policy (500 Hz).

Modellazione delle attuazione: calibrazione corrente-coppia e dinamica attuativa

Il secondo componente critico è la modellazione delle dinamiche di attuazione. Le mani destrosi usano tipicamente motori a corrente continua con riduttori a cavo, il cui comportamento devia significativamente dal modello ideale (coppia = costante × corrente) a causa di: (1) attrito dei cavi proporzionale alla curvatura del percorso, che varia con la configurazione; (2) saturazione della corrente e della velocità; (3) backlash nella trasmissione a cavi; (4) saturazione della coppia che si manifesta durante il contatto.

Il sistema calibra in simulazione la curva coppia-corrente $\tau = g(I, \dot{q}, q)$ utilizzando misure statiche e dinamiche sull'hardware reale, poi modella la deviazione dal modello nominale come perturbazione stocastica da includere nella domain randomization:

$$\tau_{real} = \tau_{nom}(I) + \Delta\tau_{friction}(\dot{q}) + \Delta\tau_{backlash}(\Delta q) + \Delta\tau_{thermal}(T) + \varepsilon \quad (26.29)$$

dove $\varepsilon \sim \mathcal{N}(0, \sigma^2_\tau)$ è il rumore residuo di coppia dopo la calibrazione dei termini deterministici. La randomizzazione di σ_τ, dei coefficienti di attrito dei cavi e del backlash, combinata con la calibrazione del modello nominale, riduce il divario attuativo a un punto in cui la policy tattile addestrata in simulazione si trasferisce zero-shot all'hardware reale.

I risultati sperimentali mostrano che la policy tattile raggiunge un errore medio di tracking della forza di presa inferiore a 0.3 N su un range operativo di 0–5 N — una precisione paragonabile a quella di un operatore umano esperto. Il reorientamento di oggetti di forme diverse (cubo, cilindro, oggetti irregolari) è dimostrato con un tasso di successo dell'82% senza fine-tuning, rispetto al 12% di una policy senza feedback tattile.

Questo risultato apre la strada all'uso del sim-to-real reinforcement learning per compiti di assemblaggio di precisione — storicamente

esclusivo appannaggio dei controllori analitici basati su modello — con la flessibilità dell'apprendimento per rinforzo nel generalizzare a oggetti e configurazioni non viste durante l'addestramento.

26.10.1 Progettazione della Ricompensa e Curriculum per la Manipolazione Destrosa

Il paradigma della presa invertita come curriculum implicito

L'innovazione metodologica più originale del sistema ByteDance Seed (2025) per la manipolazione destrosa è la configurazione di addestramento a presa invertita (inverted catching setup): la mano è tenuta capovolta con il palmo rivolto verso l'alto, e gli oggetti sono lasciati cadere dall'alto nello spazio di lavoro della mano. Questo design apparentemente bizzarro ha una motivazione ingegneristica precisa: riduce drasticamente il problema di esplorazione del RL e previene il reward hacking rispetto al paradigma tradizionale di presa da tavolo.

Nel paradigma tradizionale (mano inizializzata in posa casuale sopra un tavolo, oggetto sul tavolo), la policy deve imparare una sequenza complessa: avvicinarsi all'oggetto, orientare il palmo, chiudere le dita — una sequenza di lunghezza $H \gg 1$ che richiede un reward denso e un curriculum molto elaborato per essere appresa con RL. Nel paradigma invertito, la gravità porta l'oggetto naturalmente verso la mano, riducendo la dimensione del problema di esplorazione all'unico sotto-problema rilevante: come coordinare le dita per catturare e stabilizzare l'oggetto nell'impatto. Questo riduce l'orizzonte effettivo da $H \approx 200–500$ passi a $H \approx 50–100$ passi, con un guadagno di sample efficiency di un fattore 4–10x.

La funzione di ricompensa per il tracking della forza di presa combina tre componenti con pesi separati (eq. 1–8 del paper, riassunte qui):

$$R_{grasp}(s,a) = w_\tau \cdot R_{torque} + w_F \cdot R_{force} + w_{consist} \cdot R_{consistency} - w_{diff} \cdot R_{diff} - w_{out} \cdot R_{outer} \quad (26.45)$$

dove R_torque premia le coppie articolari che si avvicinano al target $\tau_{target} = \tau_{max} \cdot F_{cmd}$ (proporzionale al comando di forza normalizzato), R_force premia le forze di contatto all'estremo del dito vicine al target $F_{target} = F_{cmd} \cdot F_{max}$, R_consistency è un reward gaussiano che penalizza la varianza degli angoli delle falangi prossimali dei quattro diti omologhi (incoraggiando schemi di presa simili all'uomo), R_diff penalizza differenze eccessive tra le posizioni articolari delle dita adiacenti, e R_outer penalizza la deviazione delle articolazioni esterne (pollice) da un punto centrale di riferimento. La componente R_consistency è un'innovazione originale: senza di essa, il RL tende a imparare prese asimmetriche che sfruttano le specificità del simulatore (spesso una sola dita porta tutto il carico) — prese che falliscono nella realtà perché le irregolarità della superficie degli oggetti distribuiscono diversamente le forze di contatto.

Calibrazione corrente-coppia: dimostrazione del mapping non lineare

Il sistema ByteDance Seed affronta il problema della mancanza di sensori di coppia articolare nella maggior parte delle mani destrosi commerciali costruendo un mapping calibrato corrente-coppia. Il mapping nominale $\tau = k_t \cdot I$ (dove k_t è la costante di coppia del motore) è inaccurato per tre ragioni: (1) k_t dipende dalla temperatura del motore; (2) le perdite meccaniche del riduttore sono non lineari con la velocità e la posizione; (3) l'attrito statico del riduttore crea una zona morta intorno a $I = 0$ dove la coppia è zero anche con corrente non nulla.

Il mapping calibrato è costruito registrando coppie (I, $\tau_{measured}$) in condizioni quasi-statiche (velocità lenta, temperatura stabile) per un range di correnti e posizioni articolari rappresentativo. La curva di calibrazione mostra una relazione lineare a tratti con zona morta per $|I| <$

I_dead ≈ 15–20 mA e guadagno ridotto per correnti elevate (saturazione magnetica):

$$\tau\_est(I, q) = \alpha(q) \cdot sign(I) \cdot max(0, |I| - I\_dead(q)) + \beta(q) \cdot I^3 \quad (26.46)$$

dove α(q) è il guadagno di coppia dipendente dalla posizione articolare q (varia ±12% su tutto il range), I_dead(q) è la corrente di dead-zone (varia ±4 mA con la posizione per effetti di attrito), e il termine cubico $\beta(q) \cdot I^3$ modella la saturazione della coppia ad alte correnti. I coefficienti α, I_dead, β sono identificati per ciascun dito con regressione lineare su pochi secondi di dati di calibrazione. La randomizzazione di questi parametri nella simulazione (±15% su α, ±20% su I_dead) costituisce il domain randomization attuativo che garantisce il trasferimento zero-shot.

26.11 PETS, MOPO e la Frontiera Offline: Sicurezza e Trasferimento

Le sezioni precedenti hanno analizzato le strategie principali per chiudere il gap sim-to-real. È utile raccogliere queste strategie in una tassonomia coerente che le posizioni rispetto alle due dimensioni fondamentali: la modalità di raccolta dei dati (online interattivo vs. offline da dataset statico) e il meccanismo di chiusura del gap (copertura, identificazione, adattamento, architettura).

La domain randomization (sezione 26.3) appartiene alla famiglia della copertura online: si raccolgono dati da un simulatore randomizzato durante l'addestramento. MOPO (sezione 25.8) appartiene alla copertura offline: usa un dataset statico ma costruisce una ricompensa penalizzata che scoraggia la policy dall'uscire dalla regione coperta dai dati. PETS (sezione 25.9) è un metodo di adattamento online via MPC: non apprende una policy ma pianifica online con un modello ensemble, aggiornando

implicemente le credenze sui parametri attraverso la valutazione delle traiettorie simulate.

I metodi di identificazione di sistema (HALO, sezione 26.7) appartengono a un paradigma diverso: riducono il gap prima dell'addestramento della policy identificando i parametri fisici del sistema reale. Real-is-Sim (sezione 26.6) e il gemello digitale eliminano il gap durante il deployment mantenendo un simulatore sincronizzato. Il framework bi-livello (sezione 26.5) lo elimina iterativamente adattando il simulatore sulla base delle prestazioni reali della policy.

La frontiera aperta più promettente è la combinazione di questi approcci: un sistema che identifica i parametri nominali offline (stadio 1), addestrare con domain randomization adattiva centrata sul modello identificato (stadio 2), e aggiornare online il modello residuale durante il deployment (stadio 3). Questo paradigma a tre stadi è già presente embrionalmente in lavori come ASAP (Zhuang et al. 2024) e RMA (Kumar et al. 2021) per la locomozione, ma la sua estensione alla manipolazione di precisione rimane un problema aperto.

26.12 Inferenza Bayesiana Basata su Simulazione

La Simulation-Based Inference (SBI) è un framework statistico che affronta il problema dell'identificazione dei parametri del simulatore quando la funzione di verosimiglianza del simulatore è intractable — condizione che si verifica in quasi tutti i simulatori fisici realistici. L'idea centrale è di rimpiazzare la verosimiglianza esplicita $p(D|\theta)$ con un'approssimazione neurale addestrata a riconoscere quale distribuzione di parametri θ è consistente con i dati osservati D.

Formalmente, il problema di SBI è stimare il posterior bayesiano:

$$p(\theta \mid D) \propto p(D \mid \theta) \cdot p(\theta) \quad (26.32)$$

dove θ sono i parametri del simulatore (masse, attrito, parametri materiali), D = {(s_t, a_t, s_{t+1})}_t sono le traiettorie reali osservate, e p(D|θ) è la verosimiglianza delle traiettorie reali sotto il simulatore con parametri θ. Questa verosimiglianza è intractable perché richiede l'integrazione su tutti i possibili percorsi stocastici che il simulatore potrebbe generare con parametri θ.

Il metodo Sequential Neural Posterior Estimation (SNPE, Cranmer et al. 2020) risolve questo problema addestrando un stimatore neurale q_φ(θ|x) del posterior, dove x sono statistiche sommarie dei dati. L'addestramento è iterativo: (i) campiona θ ~ p_0(θ) dal prior; (ii) genera dati simulati x_sim ~ p(x|θ); (iii) addestra q_φ(θ|x) a regredire θ da x su coppie (θ, x_sim); (iv) usa q_φ come prior adattivo per la prossima iterazione. Convergenza è garantita quando il posterior approssimato q_φ converge al posterior vero p(θ|D_real), che avviene quando il numero di simulazioni è sufficiente.

L'applicazione di SBI al sim-to-real è particolarmente promettente per le seguenti ragioni. Prima: la SBI fornisce un posterior completo p(θ|D_real) anziché una stima puntuale — questo permette di usare la distribuzione di credenza sui parametri come distribuzione di domain randomization ottimale, invece di una distribuzione uniforme o gaussiana scelta a priori. Il posterior θ ~ p(θ|D_real) è già centrato sui parametri fisici consistenti con i dati reali, e la sua varianza codifica l'incertezza epistemica residua. Seconda: la SBI è applicabile anche quando i dati reali sono scarsi (poche traiettorie) attraverso le proprietà di aggiornamento bayesiano sequenziale — ogni nuova traiettoria reale restringe il posterior, riducendo progressivamente l'incertezza sui parametri.

Un risultato recente importante è la Neural Posterior Estimation applicata alla caratterizzazione del contatto (Molchanov et al. 2024): addestrando una rete neurale a stimare la distribuzione di probabilità dei parametri di attrito e di deformabilità di un oggetto sconosciuto da pochi

secondi di interazione tattile, è possibile costruire un simulatore personalizzato per quell'oggetto specifico e addestrare una policy di manipolazione in poche ore di tempo di calcolo. Questo paradigma — identify-then-train — è computazionalmente più efficiente della domain randomization ampia perché concentra il training sulla regione dello spazio dei parametri che è effettivamente plausibile per l'oggetto specifico.

26.13 Modelli di Fondazione per la Robotica: VLA e il Ruolo della Simulazione

Vision-Language-Action: i modelli di fondazione come policy gerarchiche

Un cambio di paradigma fondamentale sta avvenendo nella robotica cognitiva con l'emergere dei Vision-Language-Action model (VLA): architetture che integrano la comprensione visiva e linguistica dei modelli di fondazione (come GPT-4V, Gemini, LLaVA) con la generazione di azioni robotiche. Questi modelli sono addestrati su dataset massicci di dimostrazioni robotiche (OpenX-Embodiment: 1 milione di traiettorie da 22 robot diversi) e testi descrittivi dei compiti, e producono azioni motorie direttamente dalla specifica linguistica del compito e dall'osservazione visiva corrente.

I VLA come RT-2 (Brohan et al. 2023), OpenVLA (Kim et al. 2024) e π_0 (Black et al. 2024) condividono un'architettura comune: un encoder visivo pre-addestrato (ViT o equivalente) processa l'immagine della scena; il risultante tensore di features visive è concatenato con la codifica linguistica del comando; il modello linguistico di grandi dimensioni (7B–13B parametri) produce i token delle azioni come output autoregressivo. Formalmente:

$$a_{1:K} \sim p_\theta(a \mid v_{1:T}, l_{1:N}) \approx \Pi_{k=1}^{K} p_\theta(a_k \mid a_{<k}, enc_v(v_{1:T}), enc_l(l_{1:N})) \quad (26.33)$$

dove v_{1:T} sono i frame visivi, l_{1:N} sono i token del linguaggio naturale che descrivono il compito, e a_{1:K} sono i K valori dell'azione continua (tipicamente codificati in token discreti tramite quantizzazione uniforme o VQ-VAE).

Il ruolo della simulazione per i VLA è triplice e distinto dal ruolo tradizionale. In primo luogo, la simulazione permette la data augmentation delle dimostrazioni: partendo da una singola traiettoria reale, si generano varianti simulate (con diversi oggetti, illuminazioni, pose iniziali) che aumentano la diversità del dataset di training senza costo di raccolta dati reali. In secondo luogo, la simulazione permette la valutazione sistematica prima del deployment: poiché i VLA sono addestrati su una distribuzione ampia di compiti, è critico valutarli su compiti nuovi in modo sicuro — la simulazione fornisce questo ambiente di test. In terzo luogo, i VLA stessi possono servire come policy di alto livello per decomposizione del compito, delegando l'esecuzione a policy specializzate addestrate in simulazione: il VLA pianifica la sequenza di sub-goal («afferra l'oggetto», «portalo qui», «rilascialo») e le policy basse-livello, addestrate con sim-to-real RL per ciascun sub-goal, eseguono le azioni primitive.

Il divario sim-to-real per i modelli di fondazione

I modelli di fondazione per la robotica introducono una nuova dimensione del divario simulazione-realtà che non era presente nei sistemi tradizionali: il divario semantico. Un VLA addestrato su dataset di dimostrazioni reali ha una comprensione semantica ricca degli oggetti e dei compiti (sa cosa significa «metti la tazza sul tavolo» in contesti visivi diversi), ma questa comprensione può non corrispondere alle strutture simboliche del simulatore. Il simulatore conosce le masse, gli attriti e le geometrie degli oggetti, ma non i loro nomi o i loro ruoli funzionali nei compiti quotidiani.

Un approccio emergente per gestire questo divario semantico è la generazione procedurale di dataset simulati guidata dal VLA: il modello linguistico genera descrizioni testuali di scene e compiti («una cucina con una tazza bianca sulla sinistra del tavolo e un frutto sulla destra»), e il simulatore proceduralmente assembla la scena corrispondente, genera una policy tramite RL, e produce traiettorie di dimostrazione etichettate. Questo ciclo chiuso — VLA come supervisore semantico, simulatore come generatore di dati fisici — è il paradigma che sembra destinato a dominare la prossima generazione di sistemi robotici: il modello linguistico porta la comprensione del mondo semantico, il simulatore porta la fisica, e la policy impara a muoversi nell'intersezione tra i due.

La formalizzazione di questa pipeline è la seguente. Sia G un modello linguistico generativo e M un simulatore procedurale parametrizzato da descrizioni testuali ξ_text. Si definisce:

$$\mathcal{D}_{syn} = \{ (o_t, a_t, l) : l \sim G(l \mid task), \xi \sim M(\xi \mid l), \tau \sim \pi_{RL}(M(\xi)) \} \tag{26.34}$$

dove il dataset sintetico $\mathcal{D}_{syn}$ è generato campionando una descrizione linguistica l dal modello G, costruendo il simulatore M(ξ) corrispondente, e raccogliendo traiettorie dalla policy RL addestrata in tale simulatore. Il VLA è poi addestrato su $\mathcal{D}_{syn} \cup \mathcal{D}_{real}$, dove $\mathcal{D}_{real}$ sono le (poche) dimostrazioni reali disponibili. Il vantaggio rispetto alla domain randomization tradizionale è che la variabilità delle scene è semanticamente strutturata (lo scene generator G capisce che «tazza bianca» e «tazza rossa» sono varianti dello stesso concetto) anziché casuale (la DR tradizionale cambierebbe il colore della tazza uniformemente nell'RGB, non semanticamente).

26.14 Tassonomia Sistematica delle Soluzioni al Divario di Realtà

Ridurre il gap versus superarlo: una distinzione fondamentale

La survey di Aljalbout et al. (2026) introduce una distinzione fondamentale che organizza l'intero spazio delle soluzioni al sim-to-real gap: la distinzione tra metodi che riducono il gap (gap reduction) e metodi che lo superano (gap overcoming). I metodi della prima categoria modificano il simulatore o il processo di raccolta dati per avvicinare M_s a M_r — riducendo G_{dyn} e G_{perc}. I metodi della seconda categoria assumono un gap residuo fisso e rendono la policy robusta o adattiva a questo gap senza modificare M_s.

Questa distinzione ha implicazioni operative importanti: i metodi di riduzione del gap (system identification, residual models, real-to-sim) richiedono accesso al sistema reale prima dell'addestramento della policy, ma producono simulatori più accurati che beneficiano qualsiasi algoritmo di apprendimento. I metodi di superamento del gap (domain randomization, meta-learning, privileged information) possono essere applicati senza accesso preventivo alla realtà, ma producono policy che potrebbero non essere ottimali nel senso della sezione 26.5.

Il framework unificato suggerisce una ricetta a sei stadi per il trasferimento efficace: (1) progettare un simulatore che copre tutte le variabili rilevanti per il task; (2) ridurre il gap con system identification e residual learning; (3) applicare domain randomization attorno al modello identificato, con range calibrati; (4) addestrare la policy con architettura informata dalla fisica (privileged information, modulare); (5) valutare nel dominio reale; (6) aggiornare iterativamente il simulatore in base alle prestazioni reali. Questo ciclo converge al framework bi-livello di Anand et al. (sezione 26.5) quando le prestazioni reali sono disponibili come segnale di gradiente.

Informazione privilegiata e asimmetria attore-critico

Un pattern architetturale diventato standard nella robotica sim-to-real è l'asimmetria attore-critico con informazione privilegiata (Pinto et al. 2017; Andrychowicz et al. 2020). L'idea è che durante l'addestramento in simulazione, il critico ha accesso allo stato completo del sistema (posizioni precise degli oggetti, parametri fisici, sforzi di contatto) che non sono disponibili nella realtà, mentre l'attore ha accesso soltanto alle osservazioni che saranno disponibili nel robot reale (immagini RGB, misure propriocettive, segnali tattili). Questa asimmetria accelera l'apprendimento in simulazione (il critico con informazione completa ha una funzione di valore più precisa), senza compromettere la trasferibilità della policy (l'attore dipende soltanto da osservazioni disponibili nella realtà).

Formalmente, la funzione di valore del critico è $V_\psi(s^{priv})$ dove s^{priv} è lo stato privilegiato, mentre la policy dell'attore è $\pi_\varphi(a|o^{real})$ dove $o^{real} \subseteq s^{priv}$ è il sottoinsieme di osservazioni disponibili nella realtà. Il gradiente di policy optimization è:

$$\nabla_\varphi J(\pi_\varphi) = E_{\tau\sim\pi_\varphi} [\Sigma_t \nabla_\varphi \log \pi_\varphi(a_t|o_t^{real}) \cdot A_\psi(s_t^{priv}, a_t)] \quad (26.30)$$

dove $A_\psi(s^{priv}, a) = Q_\psi(s^{priv}, a) - V_\psi(s^{priv})$ è la funzione di vantaggio calcolata con lo stato privilegiato. Poiché il vantaggio ha varianza ridotta grazie all'informazione completa, il gradiente di policy (26.30) ha un rapporto segnale-rumore molto migliore rispetto all'alternativa simmetrica in cui l'actore-critico condividono le stesse osservazioni parziali. Durante il deployment nella realtà, soltanto l'attore π_φ è eseguito; il critico V_ψ non è necessario.

L'algoritmo Rapid Motor Adaptation (RMA, Kumar et al. 2021) estende questo framework alla stima online dei parametri ambientali: un primo modulo di imitazione apprende un encoder $z_t = \varepsilon_\varphi(\tau^{recent}_t)$ che estrae una rappresentazione compatta degli ultimi 50 passi di osservazioni-azioni reali, equivalente all'informazione privilegiata

(μ_friction, m_payload, k_contact) disponibile in simulazione. Il modulo di adattamento è addestrato per imitare il behavior di un teacher policy addestrato con informazione privilegiata, usando il policy gradient gradient sulle traiettorie reali. RMA raggiunge una performance paragonabile al teacher con informazione privilegiata dopo soli 30–50 passi di adattamento nell'ambiente reale — un risultato notevole che dimostra la potenza dell'inferenza implicita dei parametri fisici da osservazioni comportamentali.

Real-to-sim e novel view synthesis per ambienti specifici

Un'intera categoria di metodi inverte la direzione tradizionale del trasferimento: invece di addestrare in un simulatore generico e trasferire alla realtà specifica, si costruisce prima un simulatore specifico per l'ambiente target partendo da dati reali (real-to-sim), poi si addestra la policy in questo simulatore personalizzato. Il vantaggio è che il simulatore real-to-sim ha intrinsecamente $G_{dyn} \approx 0$ per l'ambiente target (per costruzione) e si può applicare domain randomization calibrata sulle incertezze fisiche residue.

Le tecnologie abilitanti per il real-to-sim sono le stesse utilizzate in Real-is-Sim (sezione 26.6): Gaussian Splatting (Kerbl et al. 2023) per la ricostruzione fotorealistica delle scene 3D da immagini multi-view, e simulatori fisici differenziabili (MuJoCo XLA, DiffTaichi, Warp) per l'identification dei parametri fisici dal movimento. La pipeline completa è: (1) scansione dell'ambiente target con telecamere calibrate; (2) ricostruzione Gaussian Splatting della scena con mesh di sfondo; (3) segmentazione degli oggetti manipolabili e identificazione delle loro proprietà fisiche (massa, attrito, rigidezza) tramite pochi esperimenti di interazione; (4) assemblaggio del simulatore personalizzato; (5) addestramento della policy; (6) deployment zero-shot.

Un risultato emblematico di questa pipeline è Torne et al. (2024), che dimostrano su task di pick-and-place in cucine reali diverse: la policy real-to-sim addestrata su uno scansione di 15 minuti del kitchen target raggiunge un tasso di successo del 78%, versus il 23% della policy addestrata su un simulatore generico. La specificità dell'ambiente compensa ampiamente il costo della scansione.

Co-addestramento con dati reali e simulati

Il co-addestramento (co-training) con dati misti simulati e reali è un paradigma crescente che sfrutta la complementarità delle due fonti: i dati simulati sono abbondanti ma imprecisi, i dati reali sono scarsi ma fedeli. Il framework formale del co-addestramento ottimizza:

$$\max_\pi \; \alpha \cdot J_{M_{sim}}(\pi) + (1-\alpha) \cdot J_{M_{real}}(\pi) \quad (26.31)$$

dove $\alpha \in [0,1]$ è un coefficiente di bilanciamento e $J_{M_{sim}}$, $J_{M_{real}}$ sono i ritorni attesi in simulazione e nella realtà. L'ottimizzazione (26.31) è approssimata raccogliendo rollout da entrambe le fonti e aggiornando la policy con una perdita pesata. Il coefficiente α viene tipicamente schedulato: si parte con α vicino a 1 (prevalenza di dati simulati quando i dati reali sono pochi) e si aumenta la quota reale man mano che la policy migliora e si accumulano dati reali.

La sfida principale del co-addestramento è il distributional shift tra i due dataset: la policy addestrata su dati simulati visita distribuzioni di stati diverse da quelle del robot reale, il che porta a un'asimmetria nel mixing del dataset. Una soluzione è il Prioritized Experience Replay (PER) applicato ai due dataset separatamente: si campionano preferenzialmente esperienze reali quando la loro priorità (errore di predizione del modello di valore) è alta, indicando che la policy non le ha ancora assimilate.

Capitolo 31 — Sintesi Finale e Prospettive

26.15 Inferenza Bayesiana sui Parametri del Simulatore

Simulation-Based Inference: dalla verosimiglianza intractable al posterior neurale

La Simulation-Based Inference (SBI), discussa da Aljalbout et al. (2026) come frontiera aperta per il sim-to-real transfer, affronta il problema fondamentale dell'identificazione bayesiana dei parametri del simulatore quando la funzione di verosimiglianza del simulatore è intractable. Il simulatore robotico Φ_sim(θ) è un programma computazionale complesso che, dati i parametri fisici θ (masse, coefficienti di attrito, elasticità), genera traiettorie simulate s^{sim}_{1:T}(θ). Invertire questo programma — inferire la distribuzione posteriore p(θ | s^{real}_{1:T}) dei parametri fisici che spiegano le traiettorie osservate nella realtà — è un problema di inferenza inversa.

Formalmente, la regola di Bayes richiede di calcolare:

$$p(\theta \mid s^{real}_{1:T}) \propto p(s^{real}_{1:T} \mid \theta) \cdot p(\theta) \quad (26.32)$$

dove p(θ) è il prior sui parametri fisici (tipicamente gaussiano centrato sui valori nominali CAD), e p(s^{real}_{1:T} | θ) è la verosimiglianza delle traiettorie reali dati i parametri θ. Il problema è che p(s^{real}_{1:T} | θ) richiede di integrare su tutte le possibili traiettorie simulate {s^{sim}_{1:T}(θ, ω)} che il simulatore potrebbe generare per il parametro θ e il rumore ω — un'integrazione su uno spazio continuo di dimensione molto alta che è genericamente intractable.

Il metodo Approximate Bayesian Computation (ABC) approssima questa integrazione con un test di vicinanza: si accetta una proposta θ ~ p(θ) se le traiettorie simulate sono sufficientemente vicine a quelle reali in termini di statistiche sommario f(·):

$$\theta \sim ABC\text{-}posterior \iff \| f(s^{sim}_{1:T}(\theta)) - f(s^{real}_{1:T}) \| \leq \varepsilon \quad (26.33)$$

dove ε è una soglia di accettazione e f è una funzione di statistiche sommario che comprime le traiettorie in vettori di features rilevanti (es. stati medi, varianze, forze di contatto medie). ABC richiede un gran numero di simulazioni (tipicamente 10^4–10^6) ed è computazionalmente costoso ma non richiede alcuna ipotesi sulla forma analitica della verosimiglianza.

L'approccio più efficiente per ambienti robotici ad alta dimensionalità è la Neural Posterior Estimation (NPE): una rete neurale q_ψ(θ | f(s^{real})) è addestrata a approssimare direttamente il posterior, usando coppie (θ^{(i)}, s^{sim}_{1:T}(θ^{(i)})) generate offline dal simulatore con parametri campionati dal prior:

$$\mathcal{L}_{NPE}(\psi) = -E_{\theta \sim p(\theta),\, s \sim \Phi_{sim}(\theta)} [\log q_\psi(\theta \mid f(s))] \quad (26.34)$$

Addestrata su N~10^5 coppie, la rete q_ψ apprende a mappare le statistiche delle traiettorie osservate al posterior sui parametri fisici. Al deployment, si osservano poche traiettorie reali, si calcolano le loro statistiche sommario f(s^{real}), e si campiona θ ~ q_ψ(· | f(s^{real})) — ottenendo una distribuzione posterior sulle proprietà fisiche reali in pochi millisecondi, senza richiedere ulteriori simulazioni. Il posterior campionato θ_1, ..., θ_K viene poi usato come distribuzione di domain randomization calibrata sui dati reali: invece di randomizzare uniformemente in un range pre-specificato, si randomizza secondo la distribuzione di incertezza residua sul sistema reale.

Il problema dell'identificabilità nel caso robotico

Un aspetto critico della SBI robotica è la non-identificabilità: dati identici possono essere generati da molteplici combinazioni di parametri fisici (es. la stessa resistenza al moto può derivare da attrito più elevato, massa maggiore, o entrambi). Questo produce posterior multi-modali che

la NPE deve catturare correttamente. Le soluzioni attuali includono l'uso di normalizing flows come architettura per q_ψ — reti neurali che modellano distribuzioni arbitrariamente complesse come composizioni differenziabili di trasformazioni semplici — e l'acquisizione sequenziale di dati (active SBI): si sceglie iterativamente il prossimo esperimento reale che massimizza la riduzione di entropia del posterior, concentrando le simulazioni sulle regioni dello spazio dei parametri dove l'incertezza è maggiore.

26.16 Representation Learning Robusto per il Trasferimento Percettivo

Il divario percettivo come problema di distribuzione visiva

Il divario percettivo G_perc, definito formalmente come E_{s~S_r}[D(O_sim(·|s) ‖ O_real(·|s))], è la componente del sim-to-real gap che affligge i sistemi di controllo visuo-motore — sistemi che mappano direttamente immagini a comandi di controllo. Anche quando la dinamica del simulatore è perfettamente calibrata, se le immagini simulate non assomigliano alle immagini reali (per differenze di illuminazione, texture degli oggetti, profondità di campo, artefatti di compressione), la policy visuo-motore addestrata in simulazione produrrà predizioni errate su osservazioni reali.

La soluzione più immediata è il domain randomization visivo: durante l'addestramento, si randomizzano colori, texture, illuminazione, e posizione della telecamera, nella speranza che la policy apprenda feature invarianti alle trasformazioni visive. Questa strategia è efficace per compiti dove le feature rilevanti sono geometriche (bordi, profondità) piuttosto che cromatiche o texturali. Tuttavia, per task che richiedono riconoscimento fine di oggetti o stima di posa basata su texture, la domain randomization visiva può essere insufficiente.

Contrastive learning per feature task-rilevanti e background-invarianti

Xing e colleghi (citati in Aljalbout et al. 2026) propongono un approccio di contrastive learning per apprendere rappresentazioni visive che sono simultaneamente task-rilevanti e background-invarianti. L'intuizione è che due osservazioni della stessa configurazione di robot e oggetto, con sfondi diversi (uno simulato, uno reale; o due scene reali con sfondi diversi), devono produrre la stessa rappresentazione latente — poiché lo sfondo non è rilevante per il controllo.

L'obiettivo di contrastive learning è la InfoNCE loss applicata a coppie (anchor, positive, negative):

$$\mathcal{L}_{NCE} = -\log \frac{\exp(\mathrm{sim}(z_a, z_p)/\tau)}{\sum_{n=1}^{N} \exp(\mathrm{sim}(z_a, z_n)/\tau)} \quad (26.35)$$

dove $z_a = \varepsilon(x_a)$ è la rappresentazione dell'ancora (es. osservazione simulata con sfondo randomizzato), $z_p = \varepsilon(x_p)$ è la rappresentazione del positivo (stessa configurazione, sfondo diverso), $z_n = \varepsilon(x_n)$ sono le rappresentazioni dei negativi (configurazioni diverse), $\mathrm{sim}(\cdot,\cdot)$ è la cosine similarity, e τ è la temperatura. La perdita (26.35) allena l'encoder a avvicinare nel latente le rappresentazioni della stessa configurazione robot-oggetto indipendentemente dallo sfondo, e ad allontanare le rappresentazioni di configurazioni diverse.

Il risultato è un encoder ε con una struttura geometrica precisa nel latente: lo spazio latente è organizzato per configurazione robot-oggetto, non per proprietà visive irrilevanti. Questo tipo di organizzazione è esattamente ciò che serve per il trasferimento sim-to-real: due osservazioni della stessa configurazione fisica — una in simulazione con texture sintetiche, una nella realtà con texture naturali — producono la stessa rappresentazione latente, e quindi la stessa azione dalla policy. La perdita (26.35) combina in modo naturale con l'obiettivo di consistenza dinamica (25.31): l'encoder allineato per dominio diventa anche dinamicamente

consistente, garantendo che le transizioni latenti apprese in simulazione siano valide nella realtà.

26.17 Regolarizzazione della Policy per il Trasferimento Robusto

Regolarizzazione della smoothness e lipschitzianità del controllo

Un aspetto spesso trascurato nella progettazione di policy per il sim-to-real transfer è la regolarità temporale e spaziale della policy: una policy che produce comandi discontinui nel tempo (alta varianza nelle azioni tra passi consecutivi) o molto sensibili a piccole perturbazioni dello stato (alta varianza spaziale) è intrinsecamente fragile al sim-to-real transfer. Le discontinuità temporali nelle azioni producono comandi meccanicamente dannosi per l'hardware (jerk elevato, picchi di coppia) e amplificano il rumore sensoriale. L'alta sensibilità spaziale significa che piccole differenze tra stato simulato e reale — inevitabili a causa del gap dinamico — producono azioni molto diverse, amplificando il performance gap.

La regolarizzazione per la smoothness temporale penalizza la variazione dell'azione tra passi consecutivi:

$$\mathcal{L}_{smooth} = E_{\tau\sim\pi}\left[\Sigma_{t=1}^{T-1} \| a_{t+1} - a_t \|^2 / \Delta t^2 \right] \quad (26.36)$$

che corrisponde alla norma L2 dell'accelerazione dell'azione nel tempo. La regolarizzazione per la lipschitzianità spaziale penalizza la sensibilità della policy alle perturbazioni dello stato:

$$\mathcal{L}_{lip} = E_{(s,\delta)\sim\pi}\left[\| \pi(s+\delta) - \pi(s) \|^2 / \|\delta\|^2 \right] \quad (26.37)$$

dove $\delta \sim \mathcal{N}(0, \sigma^2_{\delta} I)$ è una perturbazione gaussiana dello stato. La perdita (26.37) è il quadrato del rapporto di Lipschitz locale della policy e la sua minimizzazione induce una policy con comportamento robusto alle

perturbazioni dello stato — esattamente le perturbazioni che caratterizzano il gap tra stato simulato e stato reale.

Mysore e colleghi (2021) dimostrano su un drone quadricottero che la regolarizzazione di smoothness tramite perdita di funzione (invece che tramite shaping della ricompensa) è superiore: la policy addestrata con $\mathcal{L}$_smooth ha un 40% di riduzione del tasso di fallimento nel trasferimento sim-to-real rispetto alla policy addestrata senza regolarizzazione, e un 20% di riduzione rispetto alla policy con penalità di smoothness nel reward. La superiorità della loss function sulla reward penalty è spiegata dal fatto che la loss agisce direttamente sul gradiente della policy (produce una policy più regolare per costruzione), mentre la reward penalty agisce indirettamente attraverso il segnale di rinforzo (la policy può apprendere a ottimizzare la reward senza effettivamente diventare smooth, es. evitando le situazioni che attivano la penalità).

Chen e colleghi (2024) estendono la regolarizzazione di smoothness ai sistemi umanoidi, dimostrando che la combinazione di $\mathcal{L}$_smooth con la normalizzazione percentile del valore di Dreamer v3 (sezione 25.10) riduce il tasso di cadute nel trasferimento su terreni irregolari del 65% rispetto alle baseline senza regolarizzazione. La regolarità temporale del controllo è quindi tanto importante quanto la calibrazione dei parametri fisici per il trasferimento robusto — un risultato che suggerisce di trattare la regolarizzazione della policy come componente fondamentale della ricetta sim-to-real.

26.18 Simulatori Differenziabili come Strumenti di Co-Ottimizzazione

Gradients through physics: il paradigma differentiable simulation

I simulatori differenziabili — capaci di propagare gradienti attraverso l'integrazione numerica delle equazioni del moto —

rappresentano un paradigma emergente che unifica in un unico framework l'identificazione di sistema, la pianificazione e l'ottimizzazione della policy. La distinzione rispetto ai simulatori tradizionali (MuJoCo, Bullet, PhysX) è fondamentale: questi ultimi sono black box dal punto di vista dell'ottimizzazione (richiedono differenze finite per stimare i gradienti, con elevata varianza), mentre i simulatori differenziabili espongono gradienti analitici esatti attraverso ogni operazione del simulatore.

Le tecnologie abilitanti sono quattro. NVIDIA Warp applica automatic differentiation a codice GPU, producendo gradienti attraverso simulazioni fisiche con geometria arbitraria. Taichi (Hu et al. 2019) usa un linguaggio di programmazione orientato alla simulazione fisica con backward pass automatico. JAX (Bradbury et al. 2018) permette di differenziare codice Python arbitrario tramite source-to-source transformation. MuJoCo XLA (DeepMind, 2023) reimplementa MuJoCo in XLA (il backend di compilazione di JAX/TensorFlow), rendendolo completamente differenziabile ed eseguibile in parallelo su GPU.

La principale limitazione dei simulatori differenziabili è la discontinuità degli eventi di contatto: la funzione di transizione f_sim(s, a) è non differenziabile nei punti in cui un contatto inizia o termina (un oggetto tocca il suolo, un dito afferra un oggetto). Queste discontinuità causano gradienti nulli o esplosivi che destabilizzano l'ottimizzazione gradient-based. Le soluzioni attuali includono la smoothing delle forze di contatto con una funzione sigmoide:

$$f\_contact(d) = k \cdot max(0, -d) \quad \rightarrow \quad f\_smooth(d) = k \cdot log(1 + exp(-d/\tau)) \cdot \tau \quad (26.38)$$

dove d è la distanza di penetrazione tra i due corpi e τ è un parametro di smoothing che controlla il trade-off tra fedeltà del contatto ($\tau \rightarrow 0$ recupera il contatto rigido) e differenziabilità ($\tau > 0$ produce gradienti continui). Questa approssimazione è accurata per τ piccolo e consente la

propagazione di gradienti attraverso sequenze di interazioni di contatto di lunghezza fino a alcune centinaia di passi.

Co-ottimizzazione di policy, parametri fisici e progetto hardware

La disponibilità di gradienti analitici apre la strada alla co-ottimizzazione simultanea di policy e parametri fisici del sistema — un approccio che va oltre l'identificazione di sistema sequenziale (prima identifico i parametri, poi addestro la policy) verso un framework dove policy e parametri co-evolvono verso un minimo congiunto:

$$\min_{\varphi, \theta} \mathcal{L}_{task}(\pi_\varphi; \Phi_{sim}(\theta)) + \lambda \cdot \|\theta - \theta_{CAD}\|^2 \quad (26.39)$$

dove $\mathcal{L}_{task}$ è la perdita del task (negative cumulative reward), π_φ è la policy, $\Phi_{sim}(\theta)$ è il simulatore con parametri θ, e il termine di regolarizzazione penalizza la deviazione dai valori nominali del CAD. Il gradiente rispetto a φ ottimizza la policy; il gradiente rispetto a θ adatta il simulatore per rendere il task più risolvibile. Questo approccio risolve il problema dell'objective mismatch (sezione 26.5) in modo ancora più diretto del framework bi-livello: non si adatta il simulatore per essere accurato rispetto alla realtà, ma per essere ottimale per il task — una distinzione sottile ma fondamentale che, come mostrato da Nikishin et al. (2022), può produrre simulatori fisicamente inaccurati ma computazionalmente vantaggiosi per il controllo.

L'estensione più ambiziosa di questo framework è la co-ottimizzazione del design hardware: si include nella variabile di ottimizzazione anche il vettore di design del robot d (lunghezze dei link, posizioni dei giunti, configurazione degli attuatori) e si ottimizza congiuntamente policy, parametri fisici e design:

$$\min_{\varphi, \theta, d} \mathcal{L}_{task}(\pi_\varphi; \Phi_{sim}(\theta, d)) + \lambda_\theta \|\theta - \theta_{CAD}\|^2 + \lambda_d R_{design}(d) \quad (26.40)$$

dove $R_{design}(d)$ è un termine di regolarizzazione sul design che penalizza geometrie meccanicamente infattibili (link troppo corti, giunti in

conflitto) o troppo costose. Questo approccio di morphological co-optimization, esplorato da Gupta e colleghi (2021) e da Ha (2019) nel contesto della soft robotics evolutiva, produce robot co-adattati ai loro task in modo che il sim-to-real gap sia intrinsecamente ridotto: il design ottimizzato è quello per cui la policy appresa in simulazione è anche ottimale nella realtà.

26.19 Metriche Quantitative per la Valutazione del Divario Percettivo

Sim-to-Real Correlation Coefficient

Una delle lacune metodologiche più persistenti nella letteratura sim-to-real è la mancanza di metriche standardizzate per valutare se il miglioramento delle prestazioni in simulazione si traduca in miglioramento nel mondo reale. Kadian e colleghi (2020) hanno formalizzato questa questione introducendo il Sim-to-Real Correlation Coefficient (SRCC), definito come il coefficiente di correlazione di Pearson tra le prestazioni di N policy valutate in simulazione {x_1,...,x_N} e le corrispondenti prestazioni reali {y_1,...,y_N}:

$$SRCC = \Sigma_i (x_i - \bar{x})(y_i - \bar{y}) \,/\, \sqrt{[\,\Sigma_i (x_i - \bar{x})^2 \cdot \Sigma_i (y_i - \bar{y})^2\,]} \quad (26.41)$$

Un SRCC vicino a +1 indica che il simulatore è un buon predittore delle prestazioni reali: policy che si comportano meglio in simulazione si comportano meglio anche nella realtà. Un SRCC vicino a 0 indica che le prestazioni simulato-reale sono scorrelate: i miglioramenti in simulazione non hanno valore predittivo per il mondo reale. La distribuzione dei SRCC nella letteratura esistente è preoccupante: molti simulatori comunemente usati (Gazebo Classic, MuJoCo con parametri nominali) mostrano SRCC < 0.5 su task di navigazione e manipolazione, suggerendo che ottimizzare le policy sul simulatore non è equivalente a ottimizzarle per la realtà.

È importante notare che un simulatore con alto SRCC non è necessariamente più accurato fisicamente: può essere un simulatore strutturalmente semplificato che però cattura le caratteristiche più rilevanti per il controllo. Questa osservazione è coerente con il principio di equivalenza di valore (sezione 26.5): ciò che conta per il sim-to-real transfer non è la fedeltà fisica globale del simulatore, ma la sua capacità di ordinare correttamente le policy secondo le loro prestazioni reali.

Offline Replay Error

L'offline replay error è una metrica a basso costo per valutare il gap dinamico senza richiedere il deployment della policy: si registrano le traiettorie reali di un robot controllato con azioni pre-registrate, poi si replica la stessa sequenza di azioni nel simulatore e si misura la discrepanza tra le traiettorie di stato risultanti:

$$E_{replay} = (1/T) \Sigma_{t=1}^{T} \|s_t^{sim} - s_t^{real}\|_2 \quad (26.42)$$

dove s_t^{sim} è ottenuto eseguendo le azioni reali in open-loop nel simulatore. La metrica è particolarmente utile come diagnostica durante la fase di calibrazione del simulatore: un E_replay elevato indica che il modello fisico sottostante deve essere migliorato o calibrato. La limitazione di E_replay come metrica di trasferimento è che è una misura open-loop: non cattura gli effetti di feedback che possono compensare errori di modello durante il controllo in closed-loop. Una policy closed-loop può funzionare bene anche con E_replay elevato se il feedback la corregge rapidamente; al contrario, una policy feed-forward può fallire anche con E_replay basso se è sensibile a piccole perturbazioni iniziali.

Fréchet Inception Distance e analisi della fedeltà visiva

Per i sistemi di controllo visuo-motore, la metrica più rilevante per il divario percettivo G_perc è la Fréchet Inception Distance (FID), originalmente proposta da Heusel e colleghi (2017) per valutare la qualità

delle immagini generate da GAN. La FID misura la distanza tra la distribuzione delle feature visive delle immagini simulate e quella delle immagini reali nello spazio delle attivazioni di un modello Inception pre-addestrato su ImageNet:

$$FID = \|\mu_{sim} - \mu_{real}\|^2 + Tr(\Sigma_{sim} + \Sigma_{real} - 2(\Sigma_{sim}\Sigma_{real})^{1/2}) \quad (26.43)$$

dove μ_sim, μ_real sono le medie e Σ_sim, Σ_real sono le covarianze delle attivazioni Inception per i due set di immagini. La FID cattura sia le differenze di distribuzione di primo ordine (medie diverse → colori, illuminazione) sia le differenze di secondo ordine (covarianze diverse → struttura delle texture, disposizione spaziale degli elementi).

Una FID inferiore a 10 indica che le immagini simulate sono percettivamente quasi indistinguibili da quelle reali (questo livello è raggiunto da NVIDIA Isaac Sim con path tracing completo su scene semplici). Una FID tra 10 e 50 indica differenze percettibili ma task-agnostiche (la policy di controllo potrebbe funzionare comunque se le feature rilevanti sono geometriche). Una FID superiore a 100 indica differenze visive drammatiche che rendono improbabile il trasferimento di policy visuo-motrici senza domain adaptation.

La scelta tra FID e SRCC come metrica di valutazione dipende dall'obiettivo: FID misura la qualità del simulatore in senso percettivo (utile per calibrare il rendering e confrontare simulatori), mentre SRCC misura la qualità del simulatore in senso predittivo per il controllo (utile per decidere quale simulatore usare per un dato task). In generale, simulatori con FID bassa tendono ad avere SRCC alta, ma la correlazione non è perfetta: è possibile costruire un simulatore ad alta FID ma con SRCC bassa (immagini bellissime ma dinamiche sbagliate) e viceversa.

26.20 Analisi Formale della Convergenza del Framework Bi-Livello

Il Teorema di Sensitività della Stochastic Policy Gradient bi-livello

Il risultato teorico centrale del framework bi-livello di Anand et al. (2025) è il Teorema 1 della sensitività della Stochastic Policy Gradient, che deriva l'espressione analitica del gradiente del problema outer rispetto ai parametri del simulatore θ. Questo teorema — presentato qui nella sua formulazione completa per la prima volta in prosa accademica italiana — è la fondazione matematica che distingue il framework bi-livello da tutti i metodi precedenti di domain adaptation.

Il problema bi-livello è: $\max_\theta E_{\tau\sim\pi_\phi^*, P}[\Sigma_t \gamma^t r(s_t,a_t)]$ con il vincolo che $\pi_\phi^* = \text{argmax}_\phi E_{\tau\sim\pi_\phi, f_\theta}[\Sigma_t \gamma^t R_\theta(s_t,a_t)]$. La difficoltà computazionale è che il gradiente outer $\nabla_\theta J_{real}$ richiede di differenziare attraverso l'intero processo di ottimizzazione dell'inner level — una operazione che implica la differenziazione dell'argmin.

Sensitività della Catena di Markov simulata (Lemma 1)

Il Lemma 1 stabilisce che la sensitività del valore atteso di una funzione $\eta(s_k, a_k)$ sotto la catena di Markov indotta da π_ϕ nel modello f_θ rispetto al parametro θ o ϕ è:

$$\partial\, E_{s\sim\hat{\rho}^{\pi_\phi}}[\eta(s_k,a_k)] = E_{\tau}[\,\eta(s_k,a_k)\cdot\Sigma_{i=0}^{k-1}\,(\,\partial \log \pi_\phi(a_i|s_i) + \partial \log f_\theta(s_{i+1}|s_i,a_i)\,)\,] \quad (26.44)$$

dove l'operatore ð denota la derivata parziale rispetto a ϕ o θ. Questa espressione è analoga alla REINFORCE formula per il policy gradient, ma estesa alla sensitività rispetto ai parametri del modello f_θ oltre che della policy π_ϕ. Il termine $ð \log f_\theta(s_{i+1}|s_i, a_i)$ cattura l'effetto diretto della perturbazione di θ sulle transizioni generate dal simulatore.

Sensitività di Q^π e V^π (Lemma 2) e Teorema Principale

Il Lemma 2 deriva le equazioni di ricorsione per la sensitività della funzione Q rispetto ai parametri (θ o ϕ), usando le equazioni di Bellman. Per un parametro generico p ∈ {θ, ϕ}:

$$\partial_p Q^\pi(s,a) = \partial_p R(s,a) + \gamma E_{s+\sim f}[\, \partial_p V^\pi(s+) + V^\pi(s+) \cdot \partial_p \log f(s+|s,a) \,] \quad (26.45)$$

$$\partial_p V^\pi(s) = E_{a\sim\pi}[\, \partial_p Q^\pi(s,a) + Q^\pi(s,a) \cdot \partial_p \log \pi(a|s) \,] \quad (26.46)$$

Il Teorema 1 unifica questi lemmi: data la condizione di ottimalità dell'inner level $\hat{\varphi}(\phi, \theta) = 0$, la sensitività $\nabla_\theta \phi^*$ si ottiene per l'Implicit Function Theorem come:

$$\nabla_\theta \phi^* = -(\nabla_\phi \hat{\varphi})^{-1} \cdot \nabla_\theta \hat{\varphi} \quad (26.47)$$

dove la Jacobiana $\nabla_\phi \hat{\varphi}$ è la matrice di secondo ordine (Hessiana) della perdita dell'inner level rispetto ai parametri della policy — una matrice $n_\phi \times n_\phi$ la cui inversione è computazionalmente costosa ma approssimabile con il metodo del gradiente coniugato (Conjugate Gradient, CG) in $O(n_\phi)$ operazioni per passo. Il gradiente outer è poi:

$$\nabla_\theta J_{real} = E_{s\sim\rho^{\pi_\phi}}[\, \nabla_\theta \log \pi_\phi(a|s) \cdot Q^{\pi_\phi}(s,a) \,] \quad con \quad \nabla_\theta \log \pi_\phi = \nabla_\theta \phi^* \cdot \nabla_\phi \log \pi_\phi \quad (26.48)$$

La formula (26.48) è il gradiente di policy del mondo reale rispetto ai parametri del simulatore: esprime come modificare il simulatore per massimizzare le prestazioni reali della policy che emerge dall'ottimizzazione in quel simulatore. Il termine $\nabla_\theta \phi^*$ è il moltiplicatore implicito che propaga il segnale di performance reale attraverso l'intera pipeline di ottimizzazione del simulatore, dalla performance reale alla policy inner, alla sensitività dei parametri del simulatore.

In pratica, l'algoritmo bi-livello alterna: (1) esegui K passi di ottimizzazione inner (policy gradient in simulazione) per approssimare π_{ϕ^*}; (2) esegui pochi passi di rollout reale per raccogliere (s, a, s+, r); (3)

calcola il gradiente outer (26.48) usando l'approssimazione CG di (∇_ϕ φ)$^{-1}$; (4) aggiorna θ (parametri del simulatore) con un passo di gradient ascent; (5) ritorna a (1). La convergenza dell'algoritmo è garantita sotto condizioni di regolarità standard (Lipschitzianità dei gradienti, learning rate decrescente) e sotto l'ipotesi che il problema inner converga sufficientemente velocemente da fornire un'approssimazione accurata di $\nabla_\theta \phi^*$.

26.21 Covariate Shift, DAgger e la Propagazione degli Errori nell'Imitazione

Il problema del covariate shift nelle policy apprese per imitazione

Il framework di imitazione dell'MPC descritto nella sezione 26.9 soffre del problema del covariate shift (Ross e Bagnell, 2010): la policy appresa π_{DNN} è addestrata sulla distribuzione di stati $p(\xi \mid u_{MPC}, S)$ generata dall'MPC nel dominio sorgente, ma viene eseguita in un ambiente dove genera la propria distribuzione di stati $p(\xi \mid \pi_{DNN}, T)$ — potenzialmente diversa. Se π_{DNN} commette piccoli errori che portano la traiettoria fuori dalla distribuzione di addestramento, si entra in una regione dove gli errori sono più grandi (la DNN non ha mai visto questi stati), portando a errori ancora maggiori in una spirale di compounding errors.

Il divario tra le due distribuzioni è misurato dalla Total Variation distance:

$$TV\left(p(\xi|u_{MPC},S),\ p(\xi|\pi_{DNN},T)\right) \le \Sigma_{t=0}^{T-1} E_{p_t}[\ \| \pi_{DNN}(x_t) - u_{MPC}(x_t) \|_1\ ] \quad (26.47)$$

dove la disuguaglianza è il lemma di Ross e Bagnell che bound la distributional shift con la somma degli errori one-step della policy imitatrice. Questa bound implica che anche un piccolo errore one-step ε per passo produce un compounding error $O(T\cdot\varepsilon)$ sulla distribuzione degli

stati dopo T passi — spiegando perché l'imitazione dell'MPC funziona bene negli stati tipici ma degrada rapidamente in condizioni marginali o al confine dei vincoli.

DAgger: Dataset Aggregation per la correzione del covariate shift

Dataset Aggregation (DAgger, Ross et al. 2011) è l'algoritmo standard per correggere il covariate shift nell'imitazione: invece di addestrare soltanto sulla distribuzione dell'expert, si addestra iterativamente sulla distribuzione indotta dalla policy corrente. Il ciclo DAgger è:

(i) Esegui la policy corrente π_i nell'ambiente sorgente, raccogliendo traiettorie $\{\tau^{(1)}_i, ..., \tau^{(N)}_i\}$. (ii) Calcola le azioni MPC $u_{MPC}(x_t)$ per ogni stato x_t nelle traiettorie raccolte (l'MPC è l'oracolo che fornirebbe l'azione corretta). (iii) Aggiungi le coppie $(x_t, u_{MPC}(x_t))$ al dataset di addestramento. (iv) Ri-addestra π_{i+1} sull'intero dataset aggregato. La convergenza di DAgger è garantita: dopo K iterazioni, l'errore di imitazione è $O(\varepsilon_{expert} + T/\sqrt{K})$ dove ε_{expert} è l'errore di fitting del modello sui dati dell'expert.

Nel contesto del framework di Kim et al. (2026), DAgger è applicato nel dominio nominale S_{nom} (il simulatore con l'MPC nominale come oracolo): la policy DNN viene eseguita nel simulatore, gli stati visitati sono usati per calcolare nuove soluzioni MPC, e il dataset è aggregato iterativamente. Il risultato è che la distribuzione di addestramento della DNN converge alla distribuzione degli stati che la DNN stessa visita — eliminando il covariate shift per costruzione. Il framework RTMPC + DAgger + IRG ha la garanzia formale che la distribuzione degli stati visitati dalla DNN nel dominio reale T è coperta dal dataset di addestramento arricchito da DAgger, purché il gap dinamico $|S_{nom} - T| \leq \delta$ (il raggio del tubo del RTMPC).

La differenza cruciale rispetto al DAgger standard è che l'oracolo nell'iterazione i non è l'MPC eseguito sugli stati effettivi della DNN (che richiederebbe la soluzione dell'MPC in ogni stato visitato dalla DNN durante l'esplorazione), ma l'MPC nominale eseguito sugli stati propagati dal modello nominale. Questa approssimazione è valida finché la DNN resta all'interno del tubo di raggiungibilità — condizione garantita dall'IRG.

26.22 Metriche Quantitative per la Valutazione del Divario di Realtà

Il Sim-to-Real Correlation Coefficient (SRCC)

Una delle metriche più preziose introdotte nella letteratura sim-to-real è il Sim-to-Real Correlation Coefficient (SRCC), proposto da Kadian e colleghi (2020) nel contesto della navigazione e adottato dalla survey di Aljalbout et al. (2026) come standard per la valutazione della qualità predittiva dei simulatori. Il SRCC misura la correlazione di Pearson tra le prestazioni di N policy valutate in simulazione e le stesse policy valutate nella realtà:

$$SRCC = \Sigma_i\,(x_i - \bar{x})(y_i - \bar{y}) \;/\; \sqrt{[\,\Sigma_i (x_i - \bar{x})^2 \cdot \Sigma_i (y_i - \bar{y})^2\,]} \quad (26.48)$$

dove x_i è la metrica di prestazione della policy i in simulazione (es. success rate, return cumulativo), y_i è la stessa metrica nella realtà, e $\bar{x}$, $\bar{y}$ sono le medie. SRCC $\in [-1, +1]$: valori prossimi a +1 indicano che il simulatore è un ottimo proxy per la realtà (la classifica delle policy in simulazione predice la classifica nella realtà), valori prossimi a 0 indicano che il simulatore è un proxy inutile.

Il SRCC è distinto dalla performance media nella realtà: un simulatore potrebbe produrre policy con alta performance reale media ma SRCC bassa (se le policy con alta performance in sim non corrispondono a quelle con alta performance reale). In tale scenario, ogni miglioramento in

simulazione è inaffidabile come predittore di miglioramento reale — rendendo il ciclo di sviluppo cieco. Un simulatore con SRCC alta (tipicamente ≥ 0.8) permette di sviluppare e selezionare policy interamente in simulazione, riducendo le costose valutazioni sull'hardware reale.

I risultati di Kadian et al. su compiti di navigazione indoor mostrano che simulatori fotorealistici (Habitat-Sim con mesh reali) raggiungono SRCC ≈ 0.92 per la success rate di navigazione visuo-motoria, mentre simulatori con rendering semplificato raggiungono SRCC ≈ 0.45–0.60. Per la manipolazione, i dati raccolti in IndustReal mostrano SRCC ≈ 0.75–0.85 per task di inserimento di connettori con Isaac Sim fotorealistico. Questi risultati confermano che investire in rendering fotorealistico è giustificato non solo per ridurre la FID (metrica di qualità visiva), ma soprattutto per aumentare la predittività del simulatore misurata dalla SRCC.

Offline Replay Error come diagnostico del divario dinamico

L'Offline Replay Error (ORE) fornisce un diagnostico del divario dinamico senza richiedere valutazioni in loop chiuso nell'ambiente reale: si raccolgono traiettorie reali con una policy fissa, si replicano le stesse sequenze di azioni nel simulatore in loop aperto (senza feedback dal sensore reale), e si misura la divergenza tra gli stati simulati e reali:

$$E_{replay} = (1/T) \Sigma_{t=1}^{T} \| s_t^{sim}(a_{0:t-1}) - s_t^{real} \|_2^2 \quad (26.49)$$

dove s_t^{sim}(a_{0:t-1}) è lo stato del simulatore dopo aver eseguito in loop aperto le azioni reali a_{0:t-1}, e s_t^{real} è lo stato reale osservato. L'ORE misura direttamente G_dyn: se il simulatore è perfetto, ORE = 0. In pratica, ORE cresce con t a causa dell'accumulo degli errori dinamici — il tasso di crescita è un indicatore della gravità del gap dinamico.

Il profilo temporale di ORE(t) come funzione del passo t è informativo: una crescita lineare suggerisce un errore sistematico (bias del modello), una crescita esponenziale suggerisce instabilità del modello su quella traiettoria, una crescita sub-lineare (spesso osservata in task di contatto) suggerisce che il gap dinamico è principalmente dovuto ai momenti di contatto/stacco. Questo profilo può guidare la progettazione del residual model: un residual model che corregge soltanto gli stati di contatto può ridurre ORE di un fattore 3–5x con pochi parametri, mentre un residual model che corregge tutti i passi temporali richiede molti più parametri per un guadagno modesto.

31.1 Il Filo Conduttore: dalla Geometria del Corpo alla Geometria della Rappresentazione

Le sei parti di questa monografia hanno tracciato un percorso che, a una lettura retrospettiva, rivela una coerenza geometrica profonda e una progressione logica che muove dai fondamenti del corpo fisico del robot fino alle architetture predittive nel latente e alla loro validazione sperimentale. La Parte Prima ha fondato la cinematica e la dinamica del robot sulle strutture di gruppo di Lie SO(3) e SE(3), mettendo in luce come lo spazio delle configurazioni sia intrinsecamente una varietà curva non euclidea e come la correttezza degli algoritmi dipenda dal rispetto di questa curvatura. La Parte Seconda ha sviluppato la teoria degli approssimatori neurali, del reinforcement learning e dei modelli generativi, mostrando come le reti neurali profonde approssimino funzioni in spazi di Barron con tassi indipendenti dalla dimensione e come i modelli variazionali costruiscano rappresentazioni latenti strutturate. Le Parti Terza e Quarta hanno analizzato rispettivamente i world model latenti (nella famiglia RSSM-Dreamer) e l'architettura JEPA di LeCun, entrambi fondati sul paradigma della predizione nel latente come alternativa alla generazione esplicita.

Il Capitolo 25 di integrazione ha reso esplicita l'unità strutturale di queste architetture: tanto i world model quanto JEPA sono istanze del framework LPA — Latent Predictive Architecture — che si distinguono per la presenza o l'assenza del termine di ricostruzione e per il condizionamento sull'azione. Questo riconoscimento non è puramente concettuale: apre la strada all'architettura AC-JEPA, che combina la struttura causale condizionata dall'azione dei world model con l'efficienza e la semplicità delle rappresentazioni non generative di JEPA.

31.2 Simmetrie Strutturali tra le Parti

Una lettura trasversale della monografia rivela simmetrie strutturali tra concetti apparentemente distanti. Il ruolo dello Jacobiano cinematico $J(q)$ nella Parte Prima — come mappa lineare che trasforma velocità articolari in velocità operazionali — trova il suo analogo perfetto nello Jacobiano della rete neurale $J_{NN} = \partial f/\partial\theta$ nelle analisi di Neural Tangent Kernel della Parte Seconda. In entrambi i casi, le singolarità della mappa (rango deficiente di J e di J_{NN}) segnalano una perdita di informazione: nel primo caso la perdita di mobilità del robot in alcune direzioni dello spazio dei compiti, nel secondo la perdita di capacità espressiva della rete in alcune direzioni dello spazio dei parametri.

Analogamente, la funzione di Lyapunov $V(e, \dot{e})$ che garantisce la stabilità del controllo PD nella Parte Prima è strutturalmente identica all'ELBO variazionale della Parte Seconda e all'obiettivo di addestramento dell'RSSM nella Parte Terza: tutti e tre sono funzionali che bilanciano un termine di adattamento ai dati (l'errore di inseguimento, il termine di ricostruzione, la log-verosimiglianza) con un termine di regolarizzazione (la dissipazione del controllo derivativo, la divergenza KL dal prior, la divergenza KL tra posterior e prior dinamico). La struttura porta-hamiltoniana del sistema meccanico — passività, conservazione

dell'energia nella dinamica libera, dissipazione nel controllo — riappare nelle proprietà di regolarizzazione variazionale dei world model.

31.3 Le Frontiere Aperte: una Mappa dei Problemi Irrisolti

Nonostante i progressi notevoli documentati in questa monografia, le frontiere della ricerca rimangono ampie e i problemi fondamentali irrisolti sono numerosi. La causalità nelle rappresentazioni latenti — discussa nel Capitolo 26 — è forse la sfida più profonda: costruire encoder che producano rappresentazioni causalmente disentangled senza supervisione sul grafo causale è un problema di identificabilità che ha ricevuto soluzioni parziali (Khemakhem et al. 2020, Von Kügelgen et al. 2021) ma che non ha ancora una soluzione generale per l'ambito visivo non strutturato. La gap tra i teoremi di identificabilità e la praticabilità computazionale rimane enorme.

La composizionalità — la capacità di riconoscere, ragionare e pianificare su scene visive mai viste come combinazioni di concetti noti — è una proprietà che i sistemi neurali attuali possiedono solo parzialmente. I metodi oggetto-centrici discussi nel Capitolo 26 hanno compiuto progressi significativi, ma la scalabilità a scene del mondo reale con centinaia di oggetti, occlusioni dense e variabilità visiva estrema rimane un problema aperto. La connessione tra composizionalità visiva e composizionalità del ragionamento linguistico, esplorata dai modelli multimodali come GPT-4V e Gemini, suggerisce che l'integrazione di rappresentazioni visive e linguistiche possa fornire un percorso verso la composizionalità generalizzata.

Il divario simulazione-realtà, pur avendo ricevuto mitigazioni efficaci attraverso la domain randomization e i residual model, rimane un ostacolo strutturale per la robotica di precisione: in compiti che richiedono tolleranze sub-millimetriche, come l'assemblaggio di componenti elettronici, la dinamica di contatto che i simulatori non catturano è

determinante per il successo. Le soluzioni più promettenti combinano simulatori fisici ad alta fedeltà, modelli residuali apprendibili e strategie di adattamento on-line che sfruttano il feedback tattile per stimare in tempo reale i parametri di contatto.

31.4 Verso una Robotica Cognitiva Fondata sulla Predizione

La visione a lungo termine che emerge dalle architetture analizzate in questa monografia è quella di un sistema robotico in cui la percezione, la rappresentazione, la pianificazione e l'azione sono mediati da un unico substrato di predizione nel latente — una gerarchia di modelli predittivi che operano a scale temporali e spaziali diverse, ciascuno apprendendo a anticipare l'evoluzione delle rappresentazioni al livello sottostante. Questa visione, articolata da LeCun nel framework JEPA gerarchico e indipendentemente perseguita dai ricercatori di Dreamer e dei world model gerarchici, condivide radici profonde con la teoria della predictive coding di Rao e Ballard e con il free energy principle di Friston.

In questa architettura unificata, il confine tra percezione e pianificazione si dissolve: percepire significa costruire un modello predittivo della scena, e pianificare significa simulare traiettorie nel latente del modello predittivo. Il corpo del robot — con la sua geometria di Lie, le sue equazioni del moto lagrangiane, i suoi attuatori e sensori — è il substrato fisico che implementa le azioni selezionate dalla pianificazione latente e fornisce le osservazioni che alimentano l'aggiornamento del modello. La comprensione matematica di questo substrato, sviluppata nelle prime due parti di questa monografia, è la condizione necessaria — anche se non sufficiente — per costruire sistemi che non solo apprendano a muoversi nel mondo fisico, ma lo comprendano nel senso più profondo: siano in grado di anticiparne l'evoluzione, di ragionare sulle conseguenze delle proprie azioni, di adattarsi a perturbazioni impreviste attraverso un ciclo continuo di predizione, percezione e correzione.

La distanza che separa i sistemi attuali da questa visione è ancora considerevole. I world model attuali operano su orizzonti temporali di decine di secondi, su compiti con obiettivi definiti e ambienti semi-strutturati. La scalabilità a comportamenti multiminuto, a obiettivi gerarchici e a ambienti completamente aperti — il vero mondo fisico — richiede progressi fondamentali non solo nell'architettura dei modelli, ma nella comprensione teorica di cosa significhi per una macchina apprendere un modello del mondo che sia causalmente corretto, compositivamente ricco e trasferibile tra domini diversi.

Il percorso tracciato in questa monografia — dai gruppi di Lie della cinematica alle architetture predittive nel latente, passando per le teorie dell'approssimazione neurale e del reinforcement learning — è un percorso che non termina con le conclusioni di questa sintesi conclusiva. Termina, come ogni percorso intellettuale rigoroso, di fronte a una frontiera — e la frontiera, dopo sei parti che hanno percorso la geometria dei gruppi di Lie, la matematica dell'approssimazione neurale, i world model latenti, le architetture JEPA, le sfide aperte e la robotica formale, è tanto vasta quanto promettente.

31.5 Connessioni con la Neuroscienza Computazionale: Predictive Coding e Free Energy

Le architetture predittive nel latente analizzate in questa monografia non sono prive di precedenti nella neuroscienza computazionale: esse convergono verso principi teorici del funzionamento del cervello che erano stati formulati indipendentemente, con strumenti matematici diversi, dai neuroscienziati. Questa convergenza non è superficiale: suggerisce che le soluzioni architetturali trovate dall'ingegneria attraverso l'ottimizzazione empirica stanno riscoprendo principi di calcolo che l'evoluzione ha selezionato nel cervello biologico.

Il principio di predictive coding di Rao e Ballard

La teoria del predictive coding, proposta da Rao e Ballard (1999) nella sua formulazione computazionale, afferma che il cervello non elabora passivamente gli input sensoriali, ma genera attivamente predizioni dei propri input e propaga soltanto i segnali di errore di predizione (prediction error) verso i livelli gerarchici superiori. In questa teoria, ogni livello della gerarchia corticale mantiene un modello predittivo del livello inferiore; le connessioni discendenti trasmettono le predizioni del modello al livello inferiore, mentre le connessioni ascendenti trasmettono l'errore tra la predizione e l'effettivo input sensoriale.

Il formalismo matematico di Rao e Ballard è un modello variazionale a livelli. Al livello l, la rappresentazione r^l è l'attivazione corrente; la predizione del livello superiore è f^{l+1}(r^{l+1}) e l'errore di predizione è:

$$\varepsilon^l = r^l - f^{l+1}(r^{l+1}) \quad (31.1)$$

Le dinamiche dell'attivazione di r^l sono governate da:

$$\partial r^l/\partial t = -\kappa_r \varepsilon^l + \kappa_f (\partial f^l/\partial r^l)^T \varepsilon^{l-1} \quad (31.2)$$

dove il primo termine riduce l'attivazione per abbattere l'errore al livello corrente, e il secondo termine la aggiorna per ridurre l'errore al livello inferiore. Questa dinamica implementa un'inferenza variazionale gerarchica: l'evoluzione di r^l converge al posterior che minimizza simultaneamente l'errore di predizione a tutti i livelli. L'analogia con l'architettura JEPA-gerarchica (H-JEPA, equazione 25.7) è strutturalmente precisa: in JEPA, i predittori ai diversi livelli corrispondono alle funzioni f^{l+1} del predictive coding, e il meccanismo stop_grad implementa la separazione tra la predizione del livello superiore e la rappresentazione del livello inferiore che è al cuore del predictive coding.

Il Free Energy Principle di Friston

Karl Friston ha proposto il Free Energy Principle (FEP) come principio unificante del funzionamento cerebrale: il cervello minimizza continuamente la sorpresa delle proprie osservazioni sensoriali, equivalentemente massimizzando la propria model evidence ovvero il lower bound variazionale (free energy) sulla log-verosimiglianza sensoriale. La connessione con i world model è diretta: l'ELBO variazionale dell'RSSM (equazioni 25.25–25.28) è esattamente la free energy di Friston applicata a un agente artificiale che apprende un modello del mondo.

Il FEP introduce la distinzione fondamentale tra azioni perceptive (aggiornamento del modello per minimizzare la sorpresa — equivalente all'inferenza del posterior) e azioni attive (modificazione del mondo per portare le osservazioni in linea con le predizioni del modello — equivalente alla pianificazione nel latente di Dreamer). In termini RSSM, il primo corrisponde all'aggiornamento del posterior $q_\varphi(s_t | o_t)$ tramite l'encoder, il secondo corrisponde alla selezione di azioni a_t che massimizzano la ricompensa predetta $r_\psi(s_t)$ nel modello. La policy di Dreamer è quindi, nella terminologia di Friston, un active inference agent: un sistema che percepisce minimizzando la free energy e agisce scegliendo azioni che minimizzano la free energy futura prevista.

La formalizzazione di questo collegamento è la seguente. La free energy di Friston per un agente con osservazioni o, stati nascosti s e prior sulle conseguenze desiderate C(o) è:

$$F = E_{q(s|o)} [\log q(s|o) - \log p(o,s)] = KL[q \| p] - E_{q}[\log p(o|s)] \quad (31.3)$$

Per un agente robot, il prior C(o) sulle osservazioni desiderate codifica le preferenze dell'agente — gli stati del mondo che si desidera raggiungere. La minimizzazione di F rispetto alle azioni corrisponde alla

pianificazione per raggiungere tali stati preferiti. La matematica dell'active inference e quella di Dreamer sono formalmente equivalenti: la differenza principale è che l'active inference formula il problema in termini epistemologici (credenze, sorpresa, inferenza) mentre Dreamer lo formula in termini computazionali (reti neurali, gradienti, ottimizzazione). Il FEP suggerisce che i principi architetturali scoperti empiricamente nei world model — la separazione tra prior e posterior, il tradeoff tra ricostruzione e regolarizzazione KL, la pianificazione nel latente — non sono artefatti contingenti dell'ottimizzazione, ma corrispondono a principi computazionali profondi del ragionamento adattivo.

31.6 Metriche di Valutazione per il Trasferimento Sim-to-Real

Un aspetto metodologico spesso trascurato nella letteratura sperimentale è la scelta delle metriche con cui valutare il successo del sim-to-real transfer. La survey di Aljalbout et al. (2026) propone un framework di valutazione a tre dimensioni: il performance gap, il robustness gap e il sample efficiency gap. Ciascuna dimensione cattura un aspetto diverso della qualità del trasferimento.

Il performance gap G_perf (definito nella sezione 26.4) misura la differenza di ritorno atteso tra simulazione e realtà per una policy fissa. Un performance gap basso indica che la policy generalizza bene dalla simulazione alla realtà. Il robustness gap G_rob misura la varianza delle prestazioni reali su perturbazioni non coperte dalla distribuzione di addestramento:

$$G_{rob}(\pi) = Var_{\xi \notin supp(\rho)} [J_{M_r(\xi)}(\pi)] \quad (31.4)$$

Un G_rob alto indica che la policy è fragile a condizioni non viste in addestramento — un problema critico per la sicurezza dei sistemi reali. Il sample efficiency gap G_sample misura il numero di interazioni reali necessarie per chiudere il performance gap a un livello accettabile:

$$G\_sample(\pi, \varepsilon) = min\ \{\ N\_real : G\_perf(\pi\_\{N\_real\}) \leq \varepsilon\ \} \quad (31.5)$$

dove π_{N_real} è la policy dopo N_real interazioni reali (incluso fine-tuning). Un G_sample basso è fondamentale per applicazioni industriali dove le interazioni reali sono costose o pericolose. La letteratura attuale ottimizza spesso G_perf senza considerare esplicitamente G_rob e G_sample — una lacuna metodologica che contribuisce alla difficoltà di riproduzione e confronto dei risultati.

Una proposta recente (Da et al. 2025, citata in Anand et al.) è di usare il concetto di Sim2Real optimality come criterio unificante: una policy è Sim2Real optimal se J_{M_r}(π) ≥ J_{M_r}(π*) − ε, dove π* è la policy ottimale nell'ambiente reale con accesso illimitato a dati reali. Questo criterio cattura simultaneamente G_perf (la policy deve performare bene nella realtà) e implicitamente G_rob (una policy subottimale in condizioni avverse ha G_perf alto). Il framework bi-livello di Anand et al. (sezione 26.5) è il primo metodo a offrire garanzie teoriche di Sim2Real optimality sotto condizioni di regolarità.

31.7 Verso una Robotica Cognitiva Fondata sulla Predizione: Sintesi e Prospettive

Le sei parti di questa monografia, lette in retrospettiva, tracciano un percorso intellettuale che parte dalla geometria differenziale del corpo del robot — il gruppo di Lie SE(3), le equazioni di Euler-Lagrange, il controllo stabilizzante fondato sulle funzioni di Lyapunov — attraversa la matematica delle rappresentazioni neurali e del reinforcement learning, raggiunge le architetture predittive nel latente (world model, JEPA, AC-JEPA), e ritorna al dominio fisico con la Parte Sesta: pianificazione del moto su C-space, meccanica del contatto, controllo di impedenza porto-hamiltoniano, architetture VLA e validazione su benchmark. Questo percorso non è lineare, ma è guidato da un'intuizione persistente: comprendere un sistema fisico, sia esso un robot o un agente cognitivo,

significa costruire un modello interno che permetta di anticiparne l'evoluzione, di pianificare le proprie azioni — e di verificare questa comprensione nel mondo reale attraverso il contatto fisico.

Le architetture descritte nella Parte VI — i world model RSSM-Dreamer, l'architettura JEPA di LeCun, le varianti AC-JEPA e H-JEPA — sono istanze di questa intuizione realizzate a diversi livelli di astrazione e con diverse scelte di compromesso. Dreamer privilegia la fedeltà generativa (il decoder che ricostruisce le osservazioni) e la ricchezza della pianificazione (rollout immaginati con reward model separato). JEPA privilegia l'efficienza rappresentazionale (nessun decoder, predizione puramente nel latente) e la semplicità dell'obiettivo di addestramento (MSE nel latente contro target EMA). L'unificazione nel framework LPA (sezione 25.2) rivela che queste scelte non sono incompatibili ma parametrizzano uno spazio di architetture che include entrambe le famiglie come casi speciali.

Il divario simulazione-realtà, analizzato in profondità nel Capitolo 26, emerge come la sfida strutturale che condiziona ogni tentativo di trasferire le capacità apprese in simulazione al mondo fisico. La tassonomia del Capitolo 26 — da G_dyn e G_perc fino al framework bi-livello per l'ottimalità e al paradigma del gemello digitale dinamico — mostra che il problema non ha una soluzione unica: richiede invece una combinazione stratificata di approcci che agiscono a scale diverse, dal livello fisico (system identification, calibrazione dei parametri materiali) al livello algoritmico (domain randomization adattiva, gain regularization) fino al livello architetturale (informazione privilegiata, asimmetria attore-critico, Real-is-Sim).

La prospettiva unificata che emerge da questa trattazione è la seguente: il sim-to-real gap non è soltanto un problema tecnico da risolvere con più dati o simulatori migliori — è un problema epistemologico profondo che riguarda la relazione tra modelli del mondo

e mondo fisico. Un agente robotico che opera nel mondo reale deve fare continuamente inferenza sulla discrepanza tra il proprio modello interno (il world model appreso) e l'effettiva realtà — e aggiornare il modello in risposta a questa discrepanza. Questa capacità di meta-apprendimento — apprendere non soltanto a muoversi nel mondo ma a correggere il proprio modello del mondo — è ciò che distingue un sistema realmente robusto da uno che funziona soltanto nelle condizioni di test identiche a quelle di addestramento.

I sistemi descritti in questo capitolo — da HALO (identificazione di sistema differenziabile) a Real-is-Sim (gemello digitale sincronizzato) fino al framework bi-livello (adattamento del simulatore in base alle prestazioni reali) — sono tutti passi in direzione di questa capacità di meta-apprendimento. Il passo successivo, che rimane in larga misura irrisolto, è l'integrazione di questi approcci in un ciclo continuo: il robot opera nel mondo reale, raccoglie dati che rivelano la discrepanza tra il modello e la realtà, aggiorna il modello per ridurre questa discrepanza, e usa il modello aggiornato per pianificare le azioni successive. Questo ciclo — noto nella letteratura come lifelong learning o continual learning — è la frontiera verso cui converge la traiettoria della ricerca documentata in questa monografia.

La connessione con le architetture neurali di LeCun (JEPA gerarchica) e con i principi neuroscientifici di Friston (active inference, free energy principle) suggerisce che questa frontiera è percorribile: i principi computazionali che servono al cervello biologico per operare nel mondo fisico con efficienza notevole sono gli stessi che i sistemi artificiali stanno riscoprendo attraverso l'ottimizzazione su larga scala. La convergenza di questi campi — robotica, apprendimento per rinforzo, teoria delle rappresentazioni, neurosciienza computazionale — verso gli stessi principi architetturali è, forse, il segnale più promettente che siamo sulla strada giusta.

31.8 World Model come Alternativa ai Simulatori Fisici

Una delle prospettive più radicali che emerge dalla convergenza tra world model e sim-to-real transfer è la possibilità di sostituire i simulatori fisici white-box con world model neurali black-box come base per l'addestramento di policy. Questa prospettiva, discussa come frontiera aperta da Aljalbout et al. (2026) sotto la voce «Video and World Models», inverte la gerarchia tradizionale: invece di usare i world model come strumenti per il trasferimento da un simulatore fisico alla realtà, si usano i world model come simulatori nativi addestrati direttamente su dati reali, eliminando intrinsecamente il sim-to-real gap.

Il vantaggio di questo approccio è la chiusura automatica del gap percettivo e dinamico: un world model addestrato su video reali cattura esattamente la distribuzione visiva della realtà ($G_{perc} = 0$ per costruzione) e, se addestrato con sufficiente quantità di dati, apprende le dinamiche reali dell'ambiente piuttosto che un'approssimazione analitica ($G_{dyn} \to 0$ con più dati). Lo svantaggio è il compounding error delle predizioni nel latente: i world model neurali accumulano errori su orizzonti lunghi in modo difficilmente caratterizzabile, a differenza dei simulatori fisici dove gli errori sono bounded dalle proprietà analitiche del modello dinamico.

Il paradigma emergente tenta di ottenere il meglio di entrambi i mondi: si usa il simulatore fisico per generare la maggior parte dei dati di addestramento (scala illimitata, reset deterministico, parallelismo massivo), poi si affina il world model neurale su pochi dati reali per correggere il residuo del gap dinamico e percettivo. I world model neurali affinati sulle traiettorie reali sono poi usati per l'addestramento fine della policy, con interazioni reali usate soltanto per l'affinamento finale del world model — non per l'addestramento diretto della policy. Questo paradigma, denominato Sim→WM→Real pipeline, è già presente embrionalmente in lavori come TD-MPC2 (Hansen et al. 2024) e TDMPC

(Hansen et al. 2022), che usano un world model neurale nel latente addestrato on-policy come simulatore interno per l'ottimizzazione MPC.

La sfida aperta più importante per questo paradigma è la generalizzazione del world model a configurazioni dell'ambiente mai osservate durante l'addestramento: un world model addestrato in una cucina specifica può generalizzare a cucine diverse? I risultati attuali suggeriscono che world model con backbone pre-addestrati su grandi dataset visivi (es. DINOv2, CLIP) hanno una generalizzazione significativamente migliore rispetto a world model addestrati da zero, perché il backbone fornisce rappresentazioni visive che catturano struttura semantica invariante — oggetti, superfici, contenitori — indipendentemente dall'istanza specifica dell'ambiente.

31.9 Frontiere Algoritmiche: Lifelong Learning e Adattamento Continuo

La visione a lungo termine delineata nel Capitolo 31 di questa monografia — un sistema robotico che percepisce minimizzando la sorpresa del proprio modello predittivo e agisce massimizzando il ritorno previsto dalla pianificazione nel latente — richiede, per essere operativamente realizzata, una capacità che i sistemi attuali non possiedono in modo soddisfacente: la capacità di aggiornare il proprio modello del mondo in risposta a nuove esperienze, senza dimenticare le conoscenze acquisite in precedenza (lifelong learning o continual learning).

Il problema del catastrophic forgetting, documentato da McCloskey e Cohen (1989) e formalizzato da Kirkpatrick e colleghi (2017) nel contesto delle reti neurali profonde, si manifesta quando un modello viene aggiornato su nuovi dati: il gradiente di aggiornamento modifica i pesi in modo da minimizzare la perdita sui nuovi dati, ma può aumentarla sui dati precedenti se le due distribuzioni sono diverse. Per un robot che deve

imparare a operare in ambienti diversi nel corso della sua vita operativa, questo problema è strutturale: il modello addestrato in ambiente A deve poter adattarsi all'ambiente B senza dimenticare come operare in A.

Le soluzioni attuali più promettenti includono Elastic Weight Consolidation (EWC, Kirkpatrick et al. 2017), che penalizza la modifica dei pesi importanti per i task precedenti; Progressive Networks (Rusu et al. 2016), che aggiungono nuovi moduli per ogni nuovo task senza modificare i moduli precedenti; e Generative Replay (Shin et al. 2017), che usa un modello generativo per sintetizzare esempi dei task precedenti durante l'addestramento sui nuovi. Nel contesto dei world model, l'approccio Dreamer-based più promettente è il Task-Agnostic World Model (TAWM), che mantiene un world model condiviso tra task e una collezione di policy specializzate per task, con il world model aggiornato continuamente su tutte le esperienze e le policy aggiornate soltanto sulle esperienze del proprio task.

Un aspetto fondamentale della robotica cognitiva che il lifelong learning deve affrontare è la composizionalità del trasferimento: un robot che ha imparato a manipolare oggetti in una cucina dovrebbe poter trasferire queste conoscenze a una cucina diversa senza ricominciare da zero — riutilizzando le conoscenze sugli oggetti (cosa è una tazza, come si afferra) anche quando l'ambiente (disposizione degli oggetti, illuminazione, materiali) è cambiato. Questa composizionalità del trasferimento richiede che il world model mantenga rappresentazioni separate per le proprietà degli oggetti (invarianti tra ambienti) e le proprietà dell'ambiente (specifiche per ogni istanza). La sezione 26.2 ha discusso le architetture oggetto-centriche come approccio a questa separazione; l'integrazione di tali architetture con il lifelong learning è una frontiera aperta di grande rilevanza pratica.

La connessione con il problema del sim-to-real gap si chiude qui: il lifelong learning è, in un certo senso, la versione online del trasferimento

sim-to-real. Nel trasferimento sim-to-real offline, il modello apprende in simulazione e poi si trasferisce alla realtà in un unico salto. Nel lifelong learning, il modello apprende continuamente da nuove esperienze reali, aggiornando il proprio modello del mondo in modo incrementale — un processo analogo al trasferimento sim-to-real ma applicato iterativamente a ogni nuova esperienza, in un ciclo continuo di predizione, percezione, correzione e pianificazione che è la struttura computazionale fondamentale di qualsiasi sistema cognitivo adattivo.

31.10 Metriche di Correlazione Sim-to-Real e Protocolli di Valutazione

Un aspetto metodologico sistematicamente trascurato nella letteratura sperimentale è la validazione della correlazione tra prestazioni in simulazione e prestazioni nella realtà. Un sistema di apprendimento potrebbe ottenere prestazioni eccellenti in simulazione (alto ritorno atteso su episodi simulati) pur trasferendo male alla realtà — e viceversa. La mancanza di un protocollo standard per misurare questa correlazione ha reso difficile il confronto obiettivo tra approcci diversi.

La metrica Sim-to-Real Correlation Coefficient (SRCC), introdotta da Kadian et al. (2020) nel contesto della navigazione indoor, è definita come il coefficiente di correlazione di Pearson tra le prestazioni di un ensemble di N policy nella simulazione e le loro prestazioni nella realtà:

$$SRCC = \Sigma_i\,(x\_i - \bar{x})(y\_i - \bar{y}) \,/\, \sqrt{[\,\Sigma_i (x\_i - \bar{x})^2 \cdot \Sigma_i (y\_i - \bar{y})^2\,]} \quad (31.6)$$

dove x_i è la metrica di performance (success rate, ritorno cumulativo) della policy i in simulazione e y_i è la stessa metrica nella realtà. Un SRCC prossimo a +1 indica che la simulazione è un predittore affidabile delle prestazioni reali — ogni miglioramento nel simulatore si traduce in un miglioramento proporzionale nella realtà. Un SRCC prossimo a 0 indica che la simulazione non è informativa delle prestazioni

reali — il robot deve essere valutato direttamente. SRCC è distinta dalle metriche di performance assoluta: un simulatore può avere alta correlazione anche se le prestazioni reali sono sistematicamente inferiori a quelle simulate (bias) o se le prestazioni reali sono sistematicamente superiori (il che indica che il simulatore è più difficile della realtà).

L'Offline Replay Error (ORE) è una metrica complementare che non richiede il deployment della policy nella realtà: si raccolgono traiettorie reali {(s_t, a_t, s_{t+1})}, si riproducono le stesse azioni nel simulatore, e si misura la discrepanza tra la traiettoria reale e quella simulata:

$$E_{replay} = (1/T) \Sigma_{t=1}^{T} \| s_t^{sim}(a_{0:t-1}) - s_t^{real} \|_2 \quad (31.7)$$

ORE misura direttamente il divario dinamico G_dyn (sezione 26.4) lungo le traiettorie indotte dalla policy reale. È particolarmente utile nella fase di calibrazione del simulatore: un basso ORE indica che il simulatore riproduce fedelmente la fisica del sistema reale nelle condizioni operative tipiche della policy. ORE tende ad aumentare per traiettorie che coinvolgono contatti multipli e dinamiche rapide — confermando che questi sono i principali sorgenti di G_dyn nella pratica.

Per la valutazione del divario percettivo G_perc, le metriche più usate sono la Fréchet Inception Distance (FID) tra la distribuzione di immagini simulate e reali, e la t-SNE visualization degli embedding delle rappresentazioni visive: un FID alto o una separazione netta nelle proiezioni t-SNE indica che il percettore della policy riceve distribuzioni di input molto diverse in simulazione e nella realtà, il che predice un divario di prestazione elevato per policy visuo-motorie.

31.11 Il Futuro del Sim-to-Real: Simulatori Neurali e Confine con i World Model

Simulatori fisici differenziabili e il continuum con i world model

Una delle direzioni di ricerca più promettenti per il futuro del trasferimento sim-to-real è l'eliminazione della distinzione stessa tra simulatore fisico e world model neurale. I simulatori fisici differenziabili — MuJoCo XLA, Warp, DiffTaichi, PhysX con backward pass — permettono già di calcolare gradienti analitici rispetto ai parametri fisici attraverso l'intera traiettoria di simulazione, come dimostrato da HALO (sezione 26.7). Il passo successivo è integrare la differenziabilità del simulatore con la flessibilità di un modello neurale residuale: un simulatore ibrido white-box/black-box in cui la componente fisica è differenziabile e la componente neurale cattura i fenomeni non modellabili (attrito anisotropo, deformabilità dei contatti, dinamiche dei cavi).

Il modello ibrido può essere scritto formalmente come:

$$s_{t+1} = f_{physics}(s_t, a_t; \theta_{phys}) + f_{neural}(s_t, a_t, h_t; \varphi) \quad , \quad h_{t+1} = GRU(h_t, s_t, a_t) \quad (31.8)$$

dove f_physics è il simulatore fisico differenziabile (con parametri identificabili θ_phys), f_neural è un modello neurale residuale con memoria h_t, e φ sono i parametri appresi. Il gradiente dell'obiettivo di trasferimento rispetto a entrambi θ_phys e φ può essere calcolato end-to-end tramite backpropagation attraverso l'intero orizzonte di simulazione. Questo approccio è già stato dimostrato in contesti limitati (Lutter et al. 2020 per la locomozione di robot con struttura lagrangiana; Romero et al. 2024 per la manipolazione tattile), ma la sua generalizzazione a sistemi complessi multi-link con contatti multipli rimane un problema aperto.

La connessione con i world model è profonda: un simulatore ibrido differenziabile è, a tutti gli effetti, un world model strutturato che incorpora prior fisici nella propria architettura. Al contrario di RSSM e JEPA (che apprendono le dinamiche tabula rasa da dati), il simulatore ibrido ha un inductive bias fisico: sa che le traiettorie devono rispettare la

conservazione dell'energia, la terza legge di Newton, la positività della massa. Questi vincoli riducono lo spazio delle funzioni da esplorare durante l'apprendimento e migliorano la generalizzazione fuori dalla distribuzione dei dati di addestramento — esattamente il tipo di generalizzazione necessario per il trasferimento sim-to-real robusto.

Video world model come simulatori zero-shot

Un approccio alternativo che ha guadagnato attenzione recentemente è l'uso di video world model generativi come simulatori zero-shot per la robotica: invece di addestrare una policy in un simulatore fisico e trasferirla alla realtà, si addestra la policy direttamente in un video world model (come Genie 2, Sora o VideoWorldModels) che genera frame video fotorealistici condizionati sulle azioni del robot. Il vantaggio fondamentale è che il divario percettivo G_perc è intrinsecamente zero: il video world model genera immagini identiche nella distribuzione a quelle catturate da telecamere reali, poiché è addestrato su video reali.

Il limite principale dei video world model come simulatori è la coerenza fisica: un modello generativo che produce frame visivamente plausibili non garantisce che le dinamiche fisiche siano coerenti. Un robot che «spinge» un oggetto nel video world model potrebbe produrre un'immagine realistica in cui l'oggetto si muove, senza che il moto rispetti le leggi di conservazione della quantità di moto. Questo divario dinamico G_dyn^{video} è fondamentalmente diverso da G_dyn del simulatore fisico: invece di sottostimare l'attrito o sovrastimare la rigidezza, il video world model può generare dinamiche che non hanno alcuna corrispondenza con la fisica reale. La ricerca attuale su physics-informed video generation (He et al. 2024; Liu et al. 2024) cerca di incorporare prior fisici nei video world model per ridurre G_dyn^{video}, con risultati promettenti ma ancora non sufficientemente affidabili per applicazioni robotiche di precisione.

Il paradigma a lungo termine che sembra delinearsi è dunque il seguente: i simulatori fisici differenziabili (white-box, precisi ma limitati nelle dinamiche di contatto complesse) e i video world model neurali (black-box, flessibili ma fisicamente inconsistenti) sono i due estremi di uno spettro, e la frontiera della ricerca si muove verso sistemi ibridi che combinano la struttura degli uni con la flessibilità degli altri. La famiglia di architetture analizzata in questa monografia — RSSM, JEPA, AC-JEPA, i simulatori differenziabili, i framework bi-livello — sono tutti punti su questo spettro, distinti dalle scelte di dove incorporare la struttura fisica e dove lasciare libertà al modello neurale di apprendere da dati. La chiave del progresso futuro è comprendere quale struttura fisica è necessaria (non può essere appresa dai dati) e quale può essere sostituita da pattern appresi — una domanda che è al contempo ingegneristica, matematica e, in un senso profondo, epistemologica..21 Il Ciclo Real-to-Sim-to-Real: Integrazione dei Paradigmi

L'analisi dei capitoli 19 e 20 di questa monografia ha esaminato una serie di paradigmi che, visti isolatamente, sembrano approcci distinti al problema del sim-to-real transfer. A una lettura unitaria, tuttavia, questi paradigmi si rivelano elementi complementari di un ciclo iterativo che può essere formalizzato come il ciclo Real-to-Sim-to-Real (R2S2R).

Il ciclo R2S2R è composto da cinque fasi che si succedono iterativamente. La Fase 1 è l'acquisizione dei dati reali: il robot esegue una procedura di esplorazione nell'ambiente target (tipicamente vincolata per motivi di sicurezza — single-foot constraint in HALO, semplici movimenti oscillatori in Kawachi 2025), raccogliendo traiettorie di stato-azione. La Fase 2 è l'identificazione del simulatore: i dati reali sono usati per aggiornare i parametri del simulatore, tramite system identification differenziabile (HALO), Simulation-Based Inference (SBI), o ottimizzazione bi-livello (Anand et al. 2025). La Fase 3 è l'addestramento della policy: la policy è addestrata nel simulatore aggiornato, con domain

randomization calibrata attorno al modello identificato. La Fase 4 è il deployment e la valutazione: la policy viene deployata sul robot reale, raccogliendo sia le prestazioni (J_real per il segnale di gradiente outer del bi-livello) sia nuove traiettorie (per aggiornare ulteriormente l'identificazione del simulatore). La Fase 5 è l'aggiornamento iterativo: il ciclo ricomincia dalla Fase 2 con i nuovi dati reali.

La convergenza di questo ciclo — la chiave della sua utilità pratica — è garantita sotto condizioni diverse per i diversi metodi: per l'SBI con active inference, la convergenza è garantita in senso Bayesiano (il posterior converge alla distribuzione vera dei parametri con il numero di osservazioni); per il bi-livello, la convergenza è garantita sotto condizioni di regolarità al punto fisso (θ^*, π^*) tale che π^* è Sim2Real optimal; per il residual model (sezione 26.3), la convergenza è garantita se il modello residuale ha capacità sufficiente e il distributional shift tra iterazioni successive è piccolo.

Il ciclo R2S2R può essere visualizzato anche come una gerarchia di timescale: l'addestramento della policy (Fase 3) avviene alla scala di milioni di passi di simulazione (ore di GPU); l'aggiornamento del simulatore (Fase 2) avviene alla scala di decine di traiettorie reali (minuti di esperimento reale); il deployment e la raccolta di feedback (Fase 4) avviene alla scala di sessioni sperimentali (ore di esperimento). Questa separazione di scala temporale permette di parallelizzare le diverse fasi: mentre il robot raccoglie dati reali per l'aggiornamento del simulatore, la policy precedente può essere deployata e valutata; mentre il simulatore viene aggiornato, il robot può eseguire i task appresi dalla policy corrente. L'architettura computazionale distribuita del ciclo R2S2R è uno degli aspetti pratici più importanti per la sua scalabilità a sistemi robotici industriali.

31.12 Quadro Comparativo dei Metodi Sim-to-Real: Complessità, Garanzie e Applicabilità

Le sezioni 26.3–26.18 hanno presentato una varietà considerevole di approcci al sim-to-real transfer: domain randomization, identificazione di sistema differenziabile, gemello digitale dinamico, framework bi-livello, gain regularization, imitazione robusta dell'MPC, rappresentazione contrastiva, inference bayesiana sui parametri. Prima di concludere la Parte VI è utile organizzare questi approcci in un quadro comparativo che li posizioni rispetto alle dimensioni operative più rilevanti: costo di raccolta dati reali, garanzie di trasferimento, scalabilità al numero di DOF, e applicabilità a task con contact-rich physics.

La domain randomization (DR, sezione 26.3) è il metodo con il costo di dati reali più basso (zero dati reali prima del deployment) ma offre soltanto garanzie probabilistiche di copertura: la realtà deve essere nel supporto della distribuzione di randomizzazione. La DR adattiva (ADR, OpenAI 2019) elimina la necessità di specificare manualmente i range, ma richiede accesso al feedback in-loop dalla realtà per aggiustare la distribuzione. La DR condizionata ai parametri (Kawachi 2025) migliora l'efficienza della DR standard riducendo il conservatorismo attraverso la specializzazione della policy.

L'identificazione di sistema (SysID) differenziabile (HALO, sezione 26.7) richiede dati reali prima del deployment ma offre garanzie forti di riduzione di G_{dyn}: se il modello è identificabile e il simulatore è sufficientemente espressivo, $G_{dyn} \to 0$. Il costo è la raccolta di dati di esplorazione con una policy che può essere instabile (si raccolgono dati nella fase di calibrazione, non nella fase di deployment). Il framework HALO riduce questo costo con il vincolo di un piede fisso che rende l'esplorazione sicura.

Il gemello digitale dinamico (Real-is-Sim, sezione 26.6) offre teoricamente G_perf = 0 (il simulatore è sempre sincronizzato con la realtà), ma sposta il problema al meccanismo di sincronizzazione che deve essere robusto a occlusioni, oggetti non modellati e dinamiche impreviste. Il costo è la complessità ingegneristica della sincronizzazione Gaussian Splatting + simulatore fisico in tempo reale.

Il framework bi-livello (sezione 26.5) è il metodo con le garanzie teoriche più forti (Sim2Real optimality sotto condizioni di regolarità) ma richiede interazioni reali durante il deployment per aggiornare i parametri del simulatore tramite il gradiente implicito della funzione bi-livello. È computazionalmente costoso (richiede il calcolo dell'Hessiano dell'inner level) e attualmente applicabile a sistemi con dimensione moderata.

L'imitazione robusta dell'MPC (sezione 26.9) è applicabile a task con vincoli espliciti sullo stato (collision avoidance, limite di coppia) dove l'MPC è già disponibile come policy di riferimento, ma non è generalizzabile a task senza un MPC sottostante. Il suo punto di forza è la riduzione computazionale (la DNN è 10–20x più veloce dell'MPC) con garanzie di soddisfazione dei vincoli fornite dall'IRG.

L'inferenza bayesiana sui parametri (SBI, sezione 26.13) è il metodo più flessibile per identificare distribuzioni di incertezza sui parametri fisici, ma scala male alla dimensione dello spazio dei parametri per sistemi con molti DOF (il numero di simulazioni necessarie cresce esponenzialmente con la dimensione di θ per ABC, e le NPE richiedono architetture grandi e dataset di training massicci per θ di alta dimensionalità). È più indicato per sistemi con un numero limitato di parametri fisici critici (es. parametri di payload, attrito su superfici specifiche).

La representation learning contrastiva (sezione 26.14) è il metodo più efficace per ridurre G_perc per i sistemi visuo-motori: offre robustezza

alle variazioni dello sfondo e delle texture senza richiedere coppie (sim, real) corrispondenti se si usa l'approccio adversariale (25.32). Il suo limite è che non riduce G_dyn: la policy può avere rappresentazioni visive robuste ma dinamiche errate nei rollout del world model.

La ricetta ottimale per sistemi reali è la combinazione stratificata di questi approcci, ordinata per costo crescente di dati reali: (1) DR condizionata con gain regularization (zero dati reali, riduce G_dyn e G_attuativo); (2) SBI con NPE su pochi esperimenti di calibrazione (riduce G_dyn per i parametri critici); (3) representation learning contrastiva (riduce G_perc per il sistema visuo-motore); (4) aggiornamento online del residual model durante il deployment (riduce il G_dyn residuo su dati reali incrementali). Questo stack a quattro livelli è la direzione verso cui convergono i sistemi più avanzati documentati in letteratura per la manipolazione di precisione.

31.13 La Fondazione Matematica come Garanzia di Affidabilità

Questa monografia ha percorso un itinerario che, nella sua architettura complessiva, può essere letto come un argomento a favore della necessità della fondazione matematica nella robotica: non come esercizio accademico separato dalla pratica ingegneristica, ma come condizione necessaria per costruire sistemi che funzionino con affidabilità misurabile e migliorabile.

La geometria differenziale di Lie, sviluppata nella Parte I, non è ornamento: è la struttura che garantisce che gli algoritmi di cinematica e dinamica siano corretti indipendentemente dalla configurazione — che gli Jacobiani siano calcolati sul gruppo giusto, che le interpolazioni rispettino la curvatura dello spazio SO(3). Senza questa fondazione, i sistemi robotici funzionano soltanto nelle configurazioni testate, e falliscono in modo imprevedibile al di fuori di esse. I bound di approssimazione delle reti

neurali negli spazi di Barron (Parte II) non sono garanzie assolute di correttezza, ma forniscono la struttura concettuale per ragionare sulla scalabilità e la generalizzazione — per capire perché aggiungere strati oltre una certa profondità non produce miglioramenti prevedibili, o perché la dimensione del training set deve crescere con una certa potenza del numero di parametri. I risultati della Parte Sesta estendono questo quadro: la completezza probabilistica di RRT* (sezione 27.2), la convergenza del CEM in PETS (sezione 25.11), la fattibilità ricorsiva del Robust Tube MPC (sezione 26.9) e la consistenza fisica dell'identificazione dei parametri inerziali (sezione 3.3) sono tutti esempi della stessa cultura della garanzia condizionata applicata ai componenti fisici del sistema robotico.

L'ELBO variazionale dell'RSSM (dimostrato in dettaglio nella sezione 25.6), le garanzie di monotonia di MOPO (sezione 25.8), le proprietà di convergenza del CEM in PETS (sezione 25.9), i bound sul performance gap in funzione del reality gap (sezione 26.4), la convergenza del framework bi-livello verso la Sim2Real optimality (sezione 26.5) — tutti questi risultati condividono una struttura comune: forniscono condizioni sotto le quali i sistemi si comportano come progettato, e quantificano la deviazione da questo comportamento quando le condizioni sono violate. Questa struttura di garanzie condizionate è la firma matematica della robustezza: un sistema è robusto non quando funziona sempre, ma quando funziona sotto condizioni esplicitamente caratterizzate e degrada in modo prevedibile al di fuori di esse.

La sfida per la ricerca futura è estendere questa cultura della garanzia agli aspetti del sistema che attualmente mancano di fondazione teorica: la composizionalità delle rappresentazioni apprese (sezione 26.2), la causalità nelle rappresentazioni latenti (sezione 26.1), la generalizzazione dei world model a nuovi domini (sezione 31.8), l'adattamento continuo senza catastrophic forgetting (sezione 31.9). In

questi domini, la pratica è avanzata molto più velocemente della teoria — i sistemi esistenti dimostrano capacità sorprendenti che non sono ancora completamente comprese. La fondazione matematica per questi fenomeni è in costruzione, e la sua completamento è la condizione necessaria per passare da sistemi che funzionano nei laboratori a sistemi che possono essere certificati per l'uso in ambienti reali, dove la sicurezza e l'affidabilità non sono opzionali.

Il percorso della robotica cognitiva — da sistemi programmati esplicitamente a sistemi che apprendono rappresentazioni causali del mondo e pianificano nel latente — è un percorso verso l'autonomia. L'autonomia reale, però, non significa soltanto la capacità di agire senza istruzione umana continua: significa la capacità di riconoscere i propri limiti, di sapere quando il proprio modello del mondo è inaffidabile, di richiedere assistenza umana nelle situazioni che eccedono la propria competenza. Questa metacognizione — la conoscenza di cosa si sa e cosa non si sa — è forse la frontiera più lontana dalla visione delineata in questa monografia, e la più importante per una robotica che sia veramente utile, sicura e degna

Appendice A — Misura di Haar, Integrazione Invariante e Analisi Armonica su Gruppi di Lie

Il corpus matematico dei Capitoli 1, 5 e 9 di questa monografia si fonda, spesso implicitamente, sulla nozione di integrazione su gruppi di Lie e sulle strutture di analisi armonica che ne derivano. La cinematica statistica dei manipolatori ridondanti (densità dello spazio di lavoro), la stima bayesiana dell'assetto su SO(3) e il filtraggio di Kalman invariante su SE(3) richiedono tutti una misura invariante sul gruppo che permetta di definire integrali, convoluzioni e trasformate in modo geometricamente consistente. Questa appendice espone in modo rigoroso i risultati fondamentali che sono stati utilizzati o richiamati nel testo principale, integrando la trattazione con le dimostrazioni che la struttura del testo principale non ha consentito di sviluppare per intero.

A.1 Misura di Haar: Esistenza, Unicità e Forme Esplicite

Sia G un gruppo topologico localmente compatto e di Hausdorff. Una misura di Borel regolare μ su G si dice invariante a sinistra se, per ogni insieme misurabile $E \subseteq G$ e per ogni $g \in G$, si ha $\mu(gE) = \mu(E)$. Il teorema fondamentale di Haar (1933), generalizzato da von Neumann per i gruppi compatti, afferma che su ogni gruppo topologico localmente compatto esiste un'unica (a meno di costanti positive) misura di Borel regolare non identicamente nulla e invariante a sinistra, detta misura di Haar sinistra. Per i gruppi compatti — tra cui SO(3) — la misura di Haar sinistra coincide con quella destra (il gruppo è unimodulare) e può essere normalizzata in modo che $\mu(G) = 1$, restituendo una misura di probabilità sul gruppo.

Per SO(3), la misura di Haar nella parametrizzazione degli angoli di Eulero ZYZ, in cui un elemento $R \in SO(3)$ è scritto come $R(\alpha, \beta, \gamma) =$

R_z(α) R_y(β) R_z(γ) con $\alpha, \gamma \in [0, 2\pi)$ e $\beta \in [0, \pi]$, assume la forma esplicita

$$d\mu(R) = \frac{1}{8\pi^2} sin\beta \; d\alpha \; d\beta \; d\gamma \quad \text{(A.1)}$$

Il fattore sin β è il determinante jacobiano della parametrizzazione e garantisce l'invarianza: l'integrale ∫ f(R) dμ(R) produce lo stesso valore se si integra f(R) o f(R_0R) per qualsiasi $R_0 \in$ SO(3) fissa. La normalizzazione ∫_{SO(3)} dμ = 1 segue dall'identità ∫₀^{2π} dα ∫₀^{π} sinβ dβ ∫₀^{2π} dγ = $8\pi^2$.

Per SE(3), il gruppo non è compatto. La misura di Haar sinistra si fattorizza nel prodotto della misura di Haar su SO(3) e della misura di Lebesgue su $\mathbb{R}^3$:

$$d\mu_{SE(3)}(R, t) = d\mu_{SO(3)}(R) \; dt_1 \; dt_2 \; dt_3 \quad \text{(A.2)}$$

Il gruppo SE(3) è unimodulare (la misura sinistra coincide con quella destra) grazie alla struttura di prodotto semidiretto con il sottogruppo abeliano delle traslazioni. Questa proprietà semplifica considerevolmente le formule di convoluzione e trasformata, e è alla base della struttura simmetrica delle equazioni di filtering su SE(3) discusse nel Capitolo 9.

A.2 Teorema di Peter-Weyl e Decomposizione Spettrale di $L^2(SO(3))$

Il teorema di Peter-Weyl (1927) è la generalizzazione ai gruppi di Lie compatti del teorema di Fourier classico sulla retta reale: esso afferma che le funzioni di quadrato sommabile sul gruppo ammettono una decomposizione in serie ortogonale rispetto alle rappresentazioni unitarie irriducibili (UIRep) del gruppo.

Per un gruppo compatto G con misura di Haar normalizzata μ, siano {(ρ_ℓ, V_ℓ)}_l un sistema completo di UIRep a due a due inequivalenti, dove ℓ è un'etichetta discreta e d_ℓ = dim V_ℓ è la dimensione della ℓ-esima

rappresentazione. Il teorema di Peter-Weyl stabilisce che le funzioni √(d◎) ϱ◎_{mn}(g), dove ϱ◎_{mn}(g) sono i coefficienti di matrice della rappresentazione ϱ◎ in una base ortonormale, formano una base ortonormale completa di L²(G, μ). In simboli:

$$\int_G \rho\square_{mn}(g)\ \rho\square'_{m'n'}(g)\ *\ d\mu(g)\ =\ \frac{1}{d\square}\ \delta_{ll'}\ \delta_{mm'}\ \delta_{nn'}$$

(A.3)

dove δ denota il delta di Kronecker e * la coniugazione complessa. La completezza implica che ogni f ∈ L²(G) ammette l'espansione

$$f(g)\ =\ \sum_l d\square \cdot tr[f(l)\ \rho\square(g)],\quad f(l)\ =\ \int_G f(g)\ \rho\square(g)\ *\ d\mu(g) \quad \text{(A.4)}$$

con convergenza in L². La formula di Plancherel, analogo della identità di Parseval per i gruppi compatti, stabilisce:

$$\int_G |f(g)|^2\ d\mu(g)\ =\ \sum_l d\square \cdot \|f(l)\|^2_F \quad \text{(A.5)}$$

dove ‖·‖_F denota la norma di Frobenius della matrice d◎ × d◎. Le trasformate di Fourier sul gruppo (A.4) e la formula di Plancherel (A.5) sono il fondamento della teoria delle densità spaziali di lavoro dei manipolatori ridondanti: la densità di end-effector può essere espressa come convoluzione di densità per-giunto su SO(3), e tale convoluzione diventa prodotto puntuale nel dominio di Fourier.

A.3 Le Funzioni di Wigner-D: Base Armonica Esplicita su SO(3)

Per SO(3), le UIRep sono etichettate da interi o semi-interi l ∈ {0, 1/2, 1, 3/2, ...}; tuttavia, poiché SO(3) è la componente connessa dell'identità del gruppo ortogonale e ammette solo rappresentazioni a singolo valore (a differenza di SU(2) che ammette anche quelle spinoriali), le rappresentazioni irriducibili di SO(3) sono etichettate dagli interi non negativi l ∈ ℕ₀, con dimensione d◎ = 2l + 1.

I coefficienti di matrice della l-esima rappresentazione nella base standard degli armonici sferici |l, m⟩ sono le funzioni di Wigner-D:

$$D^{(l)}_{mn}(R) = \langle l, m \mid \rho^{(l)}(R) \mid l, n \rangle, \quad m, n \in \{-l, -l+1, \ldots, +l\}$$

(A.6)

Nella parametrizzazione ZYZ (α, β, γ), le funzioni di Wigner-D si esprimono come prodotto di fasi per le rotazioni attorno a z e di un fattore reale $d^{(l)}_{mn}(\beta)$ (matrice di Wigner-d ridotta) per la rotazione attorno a y:

$$D^{(l)}_{mn}(\alpha, \beta, \gamma) = e^{-im\alpha}\, d^{(l)}_{mn}(\beta)\, e^{-in\gamma} \quad \text{(A.7)}$$

dove la matrice $d^{(l)}_{mn}(\beta)$ si calcola esplicitamente mediante la formula di Wigner:

$$d^{(l)}_{mn}(\beta) = \sum_s (-1)^s \frac{\sqrt{[(l+m)!(l-m)!(l+n)!(l-n)!]}}{(l+m-s)!(l-n-s)!\, s!\, (s+n-m)!} \left(\cos\frac{\beta}{2}\right)^{2l+m-n-2s} \left(-\sin\frac{\beta}{2}\right)^{}$$

(A.8)

dove la somma su s è estesa a tutti i valori interi tali che gli argomenti dei fattoriali siano non negativi. Le relazioni di ortogonalità (A.3) diventano per SO(3):

$$\int_{SO(3)} D^{(l)}_{mn}(R)\, D^{(l')*}_{m'n'}(R)\, d\mu(R) = \frac{1}{2l+1}\, \delta_{ll'}\, \delta_{mm'}\, \delta_{nn'}$$

(A.9)

e la trasformata di Fourier su SO(3) di una funzione $f \in L^2(SO(3))$ è la famiglia di matrici (2l+1) × (2l+1):

$$\hat{f}(l) = \int_{SO(3)} f(R)\, D^{(l)}(R)^*\, d\mu(R) \in \mathbb{C}^{(2l+1)\times(2l+1)} \quad \text{(A.10)}$$

con antitrasformata $f(R) = \sum_{l=0}^{\infty} (2l+1)\, \mathrm{tr}[\hat{f}(l)\, D^{(l)}(R)]$. La proprietà fondamentale è che la convoluzione su SO(3) diventa prodotto di matrici nel dominio di Fourier: se $h = f * g$, allora $\hat{h}(l) = \hat{f}(l) \cdot \hat{g}(l)$. Questa proprietà è sfruttata da Chirikjian e colleghi (2011) per calcolare efficientemente le densità di workspace di manipolatori altamente ridondanti, dove la densità di configurazioni raggiungibili è la convoluzione di densità per-giunto.

A.4 Densità di Workspace come Convoluzione su SE(3)

L'applicazione robotica più diretta dell'analisi armonica su gruppi di Lie è il calcolo della densità di workspace per manipolatori ridondanti (Chirikjian e Kyatkin, 2001). Dato un manipolatore seriale con n giunti rotoidali, la cinematica diretta mappa ogni configurazione q ∈ (S¹)^n in una posa T(q) ∈ SE(3). La densità di workspace ϱ: SE(3) → ℝ_{≥0} è la funzione che conta (con peso uniforme rispetto alla misura di Haar sullo spazio delle configurazioni) quante configurazioni producono una posa data.

Per un manipolatore con giunti indipendenti aventi distribuzioni p_0_i sull'angolo i-esimo, la densità di workspace è la convoluzione su SE(3) delle densità per-giunto ϱ_i:

$$\rho(T) = \rho_1 * \rho_2 * \cdots * \rho\_n(T), \quad \rho\_i(T) = \int\_\{S^1\} \delta(T - T\_i(\theta))\, p_0\_i(\theta)\, d\theta \tag{A.11}$$

Nel dominio di Fourier su SE(3) — la cui teoria coinvolge le rappresentazioni unitarie del gruppo e i "Wigner-D generalizzati" per SE(3) — la convoluzione (A.11) si trasforma in prodotti di operatori. Il calcolo numerico della trasformata di Fourier su SE(3) su griglie regolari, sebbene computazionalmente pesante, può essere organizzato in modo efficiente tramite FFT sui coset SO(3) del gruppo, riducendo la complessità da O(N6) a O(N3 log N3) dove N è la risoluzione angolare. Questa tecnica è alla base degli algoritmi di Chirikjian e Kyatkin per la generazione di workspace ad alta risoluzione per manipolatori iperridondanti.

Appendice B — Control Barrier Functions: Dimostrazioni Complete

Il Capitolo 7 ha esposto la teoria delle Control Barrier Functions (CBF) in un registro prevalentemente descrittivo e applicativo, rimandando i dettagli dimostrativi per ragioni di scorrimento del testo. Questa appendice sviluppa in modo sistematico e rigoroso le dimostrazioni dei teoremi fondamentali di Ames, Xu, Grizzle e Tabuada (2017), che costituiscono il fondamento matematico del framework CBF-QP. I risultati qui dimostrati sono stati presentati nella loro forma originale nella monografia IEEE Transactions on Automatic Control; la loro comprensione profonda è indispensabile per l'applicazione critica a sistemi robotici con vincoli di sicurezza certificabili.

B.1 Funzioni di Classe K, K∞ e KL: Definizioni e Proprietà

Le funzioni di confronto sono lo strumento tecnico fondamentale per formulare condizioni di dissipatività in modo flessibile. Una funzione continua α: [0, a) → $\mathbb{R}$_{≥;0}, con a > 0 o a = +∞, appartiene alla classe K se è strettamente crescente e $\alpha(0) = 0$. Appartiene alla classe K∞ se inoltre a = +∞ e $\alpha(r) \to +\infty$ per r → +∞. Una funzione β: [0, a) × [0, +∞) → $\mathbb{R}$_{≥;0} appartiene alla classe KL se per ogni t fissato $\beta(\cdot, t) \in K$, e per ogni s fissato β(s, ·) è strettamente decrescente e tende a zero per t → +∞. Le funzioni KL caratterizzano il decadimento asintotico degli errori nei sistemi con stabilità asintotica; le funzioni K∞ compaiono nelle condizioni CBF.

Una funzione α: $\mathbb{R} \to \mathbb{R}$ appartiene alla classe K_e (classe K estesa) se è continua, strettamente crescente e $\alpha(0) = 0$ (con dominio esteso all'intera retta reale, non solo ai reali non negativi). Le funzioni di classe K_e sono la corretta generalizzazione richiesta dalle CBF zeroing, dove il valore h(x) può essere negativo per stati fuori dall'insieme safe. L'esempio canonico è

α(r) = r^3 o, più in generale, qualsiasi funzione dispari e strettamente crescente.

B.2 Zeroing Barrier Functions: Invarianza in Avanti

Si consideri un sistema di controllo affine nell'ingresso

$$\dot{x} = f(x) + g(x)u, \quad x \in \mathbb{R}^n,\ u \in U \subseteq \mathbb{R}^m \quad \text{(B.1)}$$

dove f, g sono campi vettoriali localmente Lipschitz. Sia h: $\mathbb{R}^n \to \mathbb{R}$ una funzione di classe C^1 e si definisca l'insieme safe come il sopralivello zero di h:

$$C = \{x \in \mathbb{R}^n : h(x) \geq 0\}, \quad \partial C = \{x : h(x) = 0\}, \quad Int(C) = \{x : h(x) > 0\} \quad \text{(B.2)}$$

La funzione h si dice Zeroing Barrier Function (ZBF) per il sistema (B.1) rispetto all'insieme C se esiste $\alpha \in K_e$ tale che per ogni $x \in C$:

$$\sup_{u \in U} [L_f h(x) + L_g h(x)\, u] \geq -\alpha(h(x)) \quad \text{(B.3)}$$

dove $L_f h(x) = (\nabla h(x))^T f(x)$ e $L_g h(x) = (\nabla h(x))^T g(x)$ sono le derivate di Lie di h lungo f e g rispettivamente.

Teorema B.1 (Invarianza in avanti per ZBF — Ames et al. 2017, Thm. 1)

> *Se h è una ZBF per (B.1) rispetto a C, e se u(t) è un controllo Lebesgue misurabile tale che $\dot{x}(t) = f(x(t)) + g(x(t))\, u(t)$ con $x(0) \in C$, allora $x(t) \in C$ per ogni $t \geq 0$ per cui la soluzione esiste. Ovvero, C è forward invariant per il sistema in catena chiusa.*

Dimostrazione. Si definisca la funzione ausiliaria v(t) = h(x(t)). La derivata temporale di v lungo le traiettorie del sistema è

$$\dot{v}(t) = \frac{d}{dt} h(x(t)) = \nabla h(x(t))^T \dot{x}(t) = L_f h(x(t)) + L_g h(x(t))\, u(t) \quad \text{(B.4)}$$

Per ipotesi, il controllo u(t) soddisfa la condizione ZBF, cioè L_f h(x) + L_g h(x) u(t) ≥ −α(h(x(t))). Pertanto, v(t) soddisfa l'equazione differenziale scalare

$$\dot{v}(t) \geq -\alpha(v(t)) \quad \text{(B.5)}$$

Si confronti con la soluzione dell'equazione differenziale limite $\dot{w}(t) = -\alpha(w(t))$ con condizione iniziale w(0) = v(0) = h(x(0)) ≥ 0. Poiché $\alpha \in$ K_e è strettamente crescente e $\alpha(0) = 0$, l'equazione $\dot{w} = -\alpha(w)$ ha $w \equiv 0$ come soluzione di equilibrio. Per il lemma di comparazione per disequazioni differenziali (si veda Khalil, Nonlinear Systems, Lemma 3.4), dalla (B.5) e dall'ipotesi w(0) = v(0) si ricava v(t) ≥ w(t) per ogni t ≥ 0 per cui entrambe le soluzioni esistono. Ora, se v(0) = h(x(0)) ≥ 0, allora w(0) ≥ 0 e l'equazione $\dot{w} = -\alpha(w)$ mantiene w(t) ≥ 0 per ogni t ≥ 0: se $w(t_0) = 0$ per qualche t_0, allora $\alpha(0) = 0$ implica $\dot{w}(t_0) = 0$, e l'unicità della soluzione implica w(t) = 0 per ogni $t \geq t_0$. Dunque w(t) ≥ 0 per ogni t ≥ 0, e di conseguenza v(t) = h(x(t)) ≥ w(t) ≥ 0, ovvero x(t) ∈ C per ogni t ≥ 0. □

B.3 Control Barrier Functions: Definizione e Teorema di Sicurezza

La ZBF definisce una condizione sulla derivata di h che deve essere soddisfatta da qualche ingresso u ∈ U per ogni stato x ∈ C. La Control Barrier Function estende questo concetto all'esistenza di un ingresso che soddisfa tale condizione, aprendo la strada alla sintesi di controllori. Formalmente, h è una Zeroing Control Barrier Function (ZCBF) per (B.1) se esiste $\alpha \in$ K_e tale che per ogni x ∈ Int(C) ∪ ∂C:

$$\sup_{u \in U} [L_f\, h(x) + L_g\, h(x)\, u + \alpha(h(x))] \geq 0 \quad \text{(B.6)}$$

L'insieme di tutti i controlli che soddisfano (B.6) per un dato x si denota K_{cbf}(x):

$$K_{cbf}(x) = \{u \in U : L_f\, h(x) + L_g\, h(x)\, u + \alpha(h(x)) \geq 0\} \quad \text{(B.7)}$$

Teorema B.2 (Sicurezza mediante ZCBF — Ames et al. 2017, Thm. 2)

> *Se h è una ZCBF per (B.1) rispetto a C, allora qualsiasi legge di controllo lipschitziana $u: \mathbb{R}^n \to U$ tale che $u(x) \in K_{cbf}(x)$ per ogni $x \in C$ rende l'insieme C forward invariant. Inoltre, se $x(0) \in \mathrm{Int}(C)$, le traiettorie rimangono in $\mathrm{Int}(C)$ per ogni $t > 0$.*

Dimostrazione. Per costruzione $u(x(t)) \in K_{cbf}(x(t))$, il che significa che h soddisfa la condizione ZBF lungo le traiettorie del sistema in catena chiusa. Il Teorema B.1 garantisce allora che C è forward invariant. Per la seconda parte: poiché u è Lipschitz e f, g sono localmente Lipschitz, il sistema in catena chiusa $\dot{x} = f(x) + g(x)\,u(x)$ ha un'unica soluzione massimale. Se $x(0) \in \mathrm{Int}(C)$ e per assurdo $x(t_0) \in \partial C$ per qualche primo $t_0 > 0$, allora $h(x(t_0)) = 0$ e $h(x(t)) > 0$ per $t \in [0, t_0)$. La condizione (B.7) impone $L_f h + L_g h\, u \geq -\alpha(0) = 0$ in $x(t_0)$, dunque $\dot{\gamma}(t_0) \geq 0$: ciò contraddice il fatto che h transita da valori positivi a zero, il che richiederebbe $\dot{\gamma}(t_0) \leq 0$. Per ipotesi di continuità la derivata deve essere esattamente zero in quel punto, ma allora l'unicità della soluzione con dato iniziale $h(x(t_0)) = 0$ e $\dot{\gamma}(t_0) = 0$ implicherebbe che la soluzione rimanga identicamente sul bordo ∂C, contraddicendo $h(x(t)) > 0$ per $t \in [0, t_0)$. La contraddizione mostra che $x(t) \in \mathrm{Int}(C)$ per ogni $t > 0$. □

B.4 Unificazione CLF-CBF in un Programma Quadratico

La forza del framework di Ames et al. (2017) è la possibilità di mediare sicurezza (CBF) e performance/stabilità (CLF) in un unico problema di ottimizzazione convessa risolvibile in tempo reale. Una Control Lyapunov Function (CLF) $V: \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ per il sistema (B.1) soddisfa l'esistenza di $c_1, c_2 > 0$ e una funzione definita positiva γ tali che $c_1 \|x\|^2 \leq V(x) \leq c_2 \|x\|^2$, e per ogni $x \neq 0$:

$$\inf_{u \in U} \left[L_f V(x) \;+\; L_g V(x)\, u \;+\; \gamma(V(x)) \right] \leq 0 \qquad \text{(B.8)}$$

Il programma quadratico CLF-CBF che media sicurezza e stabilità è:

$$u^*(x) = \underset{(u,\delta)\in\mathbb{R}^{m+1}}{\operatorname{argmin}} \frac{1}{2} u^T H u + p^T u + \frac{p}{2}\delta^2 \quad \text{(B.9)}$$

soggetto ai vincoli

$$L_f V(x) + L_g V(x)\, u + \gamma(V(x)) \le \delta \quad (CLF,\ relaxed) \quad \text{(B.10a)}$$

$$L_f h(x) + L_g h(x)\, u + \alpha(h(x)) \ge 0 \quad (CBF,\ hard) \quad \text{(B.10b)}$$

dove $H \succeq 0$ è una matrice di pesi semi-definita positiva, $p \in \mathbb{R}^m$ è un termine lineare (tipicamente derivato da un controllore nominale che si vuole seguire), e $p > 0$ è una penalizzazione della variabile di rilassamento $\delta \in \mathbb{R}$. Il vincolo CLF viene rilassato tramite δ per garantire la fattibilità quando le due condizioni sono in conflitto: in tal caso la sicurezza (B.10b) è preservata in modo hard, mentre la convergenza verso l'obiettivo può essere temporaneamente rallentata. Il QP (B.9)–(B.10) ha la forma di un programma quadratico convesso con vincoli lineari, risolvibile in $O(m^3)$ tramite algoritmi attivi standard, ed è di dimensione indipendente dalla dimensione dello stato n — la proprietà chiave che lo rende implementabile in tempo reale su hardware embedded per il controllo di robot.

La fattibilità del QP per ogni $x \in C$ è garantita dall'esistenza di almeno un $u \in U$ che soddisfi (B.10b), che è esattamente la condizione di ZCBF (B.6). La Lipschitz-continuità della soluzione $u^*(x)$ rispetto a x, che garantisce l'esistenza di soluzioni al sistema in catena chiusa, è dimostrata da Ames et al. (2017) attraverso l'analisi della funzione di valore del QP parametrico e la regolarità della sua soluzione al variare di x, usando risultati di programmazione sensibility analysis per QP (Bank et al., 1983).

Appendice C — Distribuzioni Gaussiane su Gruppi di Lie e Filtraggio Invariante

Il Capitolo 9 ha introdotto la stima bayesiana dell'assetto su varietà di Lie, presentando il filtro di Kalman invariante come estensione geometricamente corretta del filtro EKF ai sistemi evolutivi su SE(3). Questa appendice sviluppa i fondamenti matematici sottostanti, con particolare attenzione alla nozione di distribuzione gaussiana su un gruppo di Lie non-euclideo e all'equazione del calore che ne governa la propagazione temporale.

C.1 L'Equazione del Calore su SO(3) e la Distribuzione Gaussiana

La generalizzazione della distribuzione gaussiana a SO(3) prende la forma della soluzione fondamentale dell'equazione del calore sul gruppo, dove il laplaciano euclideo Δ_{ℝ^n} viene sostituito dall'operatore di Laplace-Beltrami Δ_{SO(3)} associato alla metrica bi-invariante. L'operatore Δ_{SO(3)}, espresso in coordinate esponenziali tramite i generatori {E_1, E_2, E3} dell'algebra di Lie so(3), è:

$$\Delta_{SO(3)} = E_1^2 + E_2^2 + E3^2 \quad \text{(C.1)}$$

dove E_i agisce sulle funzioni su SO(3) come derivata direzionale invariante a sinistra nella direzione del generatore i-esimo. L'equazione del calore su SO(3) per una densità di probabilità p: SO(3) × [0,+∞) → ℝ_{≥0} è:

$$\partial_t\, p(R,\, t) = \frac{D}{2}\, \Delta_{SO(3)}\, p(R,\, t) \quad \text{(C.2)}$$

dove D > 0 è il coefficiente di diffusione. La soluzione fondamentale con condizione iniziale p(R, 0) = δ(R, I) si ottiene tramite la decomposizione di Fourier su SO(3) illustrata nell'Appendice A: poiché E_i agisce come

operatore di Casimir sulle rappresentazioni irriducibili, ogni matrice D◉(R) è autofunzione di Δ_{SO(3)} con autovalore −l(l+1):

$$\Delta_{SO(3)}\, D^{(l)}_{mn}(R) \;=\; -\, l(l+1)\, D^{(l)}_{mn}(R) \quad \text{(C.3)}$$

Applicando la trasformata di Fourier (A.10) a (C.2) e usando (C.3), l'equazione del calore si trasforma nell'equazione ordinaria p̃̇(l, t) = −(D/2) l(l+1) p̃(l, t), con soluzione p̃(l, t) = p̃(l, 0) e^{−(D/2) l(l+1) t}. L'antitrasformata restituisce la soluzione fondamentale esplicita:

$$p(R,\, t) \;=\; \sum_{l\,=\,0}^{\infty} (2l+1)\, e^{-\frac{D}{2}\, l(l+1)\, t}\, \mathrm{tr}[D^{(l)}(R)] \quad \text{(C.4)}$$

Questa funzione è la "gaussiana" su SO(3) centrata nell'identità, o Gaussiana di Lie, e gioca il ruolo della distribuzione e^{−‖x‖²/(2σ²)} sulla retta reale. Per t piccolo la distribuzione è concentrata intorno all'identità, mentre per t → +∞ converge alla misura di Haar uniforme. Una Gaussiana centrata in $R_0 \neq I$ si ottiene tramite traslazione sinistra: $p(R; R_0, t) = p(R_0^{-1}R, t)$.

C.2 La "Banana Distribution" in Coordinate Esponenziali

Un risultato sorprendente e controintuitivo è dimostrato da Long, Wolfe, Mashner e Chirikjian (RSS 2012): la distribuzione di probabilità della posa di un end-effector il cui errore di posizione nei giunti è gaussiano diventa gaussiana anche nello spazio delle pose SE(3) se misurata in coordinate esponenziali, almeno per distribuzioni sufficientemente concentrate. Questo risultato, noto come "the Banana Distribution is Gaussian", è fondamentale per la progettazione di filtri di Kalman su SE(3).

Più precisamente: sia $X = \exp(\xi) \in SE(3)$ un elemento del gruppo con $\xi \in$ se(3) variabile aleatoria gaussiana con media ξ‰ e covarianza $\Sigma \in \mathbb{R}^{6\times 6}$. La distribuzione di X in SE(3) non è gaussiana nella rappresentazione matriciale, ed è la "banana distribution" osservata nelle proiezioni sulla traslazione: la traiettoria di una distribuzione di pose in

SE(3) appare "arcuata" quando proiettata su $\mathbb{R}^3$, poiché la componente rotazionale accoppia angoli e traslazioni. Tuttavia, nelle coordinate esponenziali $\xi = \log(X_0^{-1} X)$, dove $X_0 = \exp(\xi‰)$ è la posa media, la distribuzione dell'errore ξ è approssimativamente gaussiana per Σ piccola:

$$p(\xi) \approx N(0, \Delta\Sigma), \quad \Delta\Sigma = \Sigma + O(\|\Sigma\|^2), \quad \xi = log(X_0^{-1} X) \quad \text{(C.5)}$$

L'approssimazione (C.5) diventa esatta al primo ordine in Σ e le correzioni del secondo ordine coinvolgono le strutture di curvatura del gruppo (i termini di Lie bracket dell'algebra). Questo risultato giustifica l'uso della linearizzazione nell'algebra di Lie come sostituto delle operazioni euclidee nei filtri di Kalman, e formalizza il motivo per cui il filtro di Kalman invariante su SE(3) è più accurato dell'EKF standard nelle parametrizzazioni quaternioniche o di Eulero: l'errore di linearizzazione è del secondo ordine in Σ nelle coordinate di Lie, mentre nella parametrizzazione di Eulero compaiono termini di primo ordine legati alla singolarità di rappresentazione.

C.3 Il Filtro di Kalman Invariante (IEKF) su SE(3)

Il filtro di Kalman invariante a destra (right-invariant IEKF) opera sullo stato $X \in SE(3)$ definendo l'errore come la perturbazione destra $\xi = \log(\hat{X}^{-1} X) \in se(3)$ tra la stima $\hat{X}$ e lo stato vero X. Le equazioni del filtro mantengono la struttura invariante rispetto al cambiamento di sistema di riferimento.

Il passo di predizione propaga la stima $\hat{X}$ e la covarianza Σ_ξ dell'errore secondo

$$\dot{\hat{X}} = f(\hat{X}, u), \quad \dot{\Sigma}\_\xi = F \Sigma\_\xi F^{\mathrm{T}} + Q \quad \text{(C.6)}$$

dove $F = (\partial \log(\hat{X}^{-1} f(X, u)) / \partial\xi)|\_{\xi=0}$ è lo Jacobiano del modello di processo in coordinate di Lie e Q è la covarianza del rumore di processo. Il

passo di aggiornamento, data una misura y = h(X) + n, calcola il guadagno di Kalman e corregge la stima:

$$K = \Sigma\_\xi H^{\mathrm{T}} (H \Sigma\_\xi H^{\mathrm{T}} + R)^{\{-1\}}, \quad \xi^{\cdot} = K (y - h(\hat{X})), \quad \hat{X} \leftarrow \hat{X}\, exp(\xi^{\cdot})$$

(C.7)

$$\Sigma\_\xi \leftarrow (I - KH)\, \Sigma\_\xi \quad \text{(C.8)}$$

dove H = (∂ h(X exp(ξ)) / ∂ξ)|_{ξ=0} è lo Jacobiano del modello di osservazione in coordinate di Lie e R è la covarianza del rumore di misura. La correzione finale $\hat{X} \leftarrow \hat{X}\exp(\xi^{\cdot})$ è un'operazione sul gruppo (non una somma vettoriale), garantendo che la stima rimanga sempre in SE(3). La proprietà fondamentale dell'IEKF è la consistenza geometrica: l'errore ξ evolve nell'algebra di Lie se(3), uno spazio vettoriale euclideo di dimensione 6, e le operazioni di aggiornamento sono esattamente quelle del filtro di Kalman standard in tale spazio vettoriale.

Appendice D — Linearità nei Parametri Dinamici e Regressor del Manipolatore

Il Capitolo 3 ha enunciato la proprietà fondamentale di linearità nei parametri del modello dinamico dei manipolatori, sfruttata dal controllore adattativo di Slotine-Li. Questa appendice dimostra in modo sistematico la struttura del regressor Y(q, q̇, q̇_r, q̈_r) e caratterizza il vettore minimale dei parametri identificabili.

D.1 Il Tensore di Inerzia come Forma Bilineare e la Base dei Parametri Inerziali

I parametri inerziali di ogni link i del manipolatore sono completamente caratterizzati da dieci quantità scalari: la massa m_i ∈ ℝ_{>0}, il primo momento di massa (momento statico) m_i c_i ∈ ℝ³ dove c_i è la posizione del centro di massa nel frame del link, e i sei parametri indipendenti del tensore di inerzia I_i rispetto all'origine del frame del link. Il tensore di inerzia è la forma bilineare simmetrica definita positiva su ℝ³:

$$I\_i \;=\; \int\_{B\_i} \rho(p)\,(\|p\|^{2\,\mathrm{T}} I \;-\; p\,p^{\mathrm{T}})\,dV, \quad I\_i \;=\; [I\_\{xx\}\; I\_\{xy\}\; I\_\{xz\};\; I\_\{xy\}\; I\_\{yy\}\; I\_\{yz\};\; I\_\{$$

(D.1)

dove B_i è il volume del link i, ϱ(p) è la densità di massa e la formula $\|p\|^2 I - p\,p^T$ è il tensore di Huygens-Steiner. Il vettore dei parametri inerziali del link i è:

$$\pi\_i \;=\; (m\_i,\; m\_i\, c\_\{ix\},\; m\_i\, c\_\{iy\},\; m\_i\, c\_\{iz\},\; I\_\{xx,i\},\; I\_\{xy,i\},\; I\_\{xz,i\},\; I\_\{yy,i\},\; I\_\{yz,i\},\; I\_\{$$

(D.2)

Il vettore globale dei parametri del manipolatore a n link è θ = $(\pi_1^T, \ldots, \pi\_n^T)^T \in \mathbb{R}^\{10n\}$. Il teorema di linearità nei parametri afferma che le equazioni di Newton-Euler, dopo le opportune proiezioni, si possono scrivere come:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = Y(q, \dot{q}, \dot{q}\_r, \ddot{q}\_r)\,\theta \quad \text{(D.3)}$$

dove Y ∈ ℝ^{n×10n} è la matrice regressore. La linearità (D.3) è non banale: M(q) e C(q, q̇) dipendono in modo polinomiale da sin(q_i) e cos(q_j), che appaiono come funzioni altamente nonlineari delle configurazioni, eppure quando si esprimono M, C, g tramite la decomposizione ricorsiva Newton-Euler (Featherstone, 2008), ogni termine risulta lineare nei parametri π_i del corrispondente link. La dimostrazione procede per induzione sulla catena cinematica: nel passo base, il singolo link rigido ha energia cinetica $T = (1/2)\,\omega^T I\,\omega + (1/2)\,m\,v\_c^T\,v\_c = (1/2)\,\xi^T M\_\{\}(q)\,\xi$ che è quadratica in ξ e lineare in (m, mc, I); nel passo induttivo, la composizione di trasformazioni omogenee di SE(3) e la proiezione sulle coordinate articolari preservano questa linearità.

D.2 Parametri Identificabili e Forma Minimale del Regressor

Non tutti i 10n parametri in θ sono identificabili individualmente da misure di coppia e accelerazione: esistono combinazioni lineari di parametri che producono lo stesso contributo alle coppie per qualsiasi traiettoria q(t). Il rango della matrice regressore Y, valutato su una traiettoria persistentemente eccitante, determina il numero di parametri identificabili. Sia $W = \int_0^T Y^T(q(t), \ldots)\,Y(q(t), \ldots)\,dt$ la matrice di informazione di Fisher; il numero di parametri identificabili è rango(W), e vale al più 10n ma tipicamente è inferiore per robot con struttura speciale (per esempio, se due giunti consecutivi sono paralleli, certi prodotti di massa-distanza non contribuiscono separatamente). La forma minimale del regressor $Y\_r \in \mathbb{R}^{n\times p}$ con p = rango(W) si ottiene tramite la scomposizione QR della matrice W e la proiezione del vettore θ sullo spazio immagine di W^T.

Bibliografia

Le voci bibliografiche sono organizzate per ordine alfabetico del primo autore. Per i lavori citati con anno e autore nel testo, la corrispondenza è stabilita dal cognome e dall'anno di pubblicazione.

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas-Xirgo, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Wierstra, D., Wiltschko, D., Zaremba, W. (2019). Solving Rubik's Cube with a Robot Hand. arXiv:1910.07113.

Ames, A.D., Xu, X., Grizzle, J.W., Tabuada, P. (2017). Control Barrier Function Based Quadratic Programs for Safety Critical Systems. IEEE Transactions on Automatic Control, 62(8), 3861–3876.

Andrychowicz, O.M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Wierstra, D., Zaremba, W. (2020). Learning Dexterous In-Hand Manipulation. The International Journal of Robotics Research, 39(1), 3–20.

Barron, A.R. (1993). Universal Approximation Bounds for Superpositions of a Sigmoidal Function. IEEE Transactions on Information Theory, 39(3), 930–945.

Bengio, Y., Courville, A., Vincent, P. (2013). Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1798–1828.

Black, K., Brown, N., Driess, D., Du, Y., Espeholt, L., Florence, P., Hejna, D., Finn, C., Hafner, D., Hausman, K. (2024). π_0: A Vision-Language-Action Flow Model for General Robot Control. arXiv:2410.24164.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al. (2023). RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. arXiv:2307.15818.

Chi, J., Feng, Z., Du, Z., Xu, R., Mo, K., Guibas, L. (2023). Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. Proceedings of Robotics: Science and Systems (RSS).

Chirikjian, G.S. (2011). Stochastic Models, Information Theory, and Lie Groups, Volume 1: Classical Results and Geometric Methods. Birkhäuser, Boston.

Chirikjian, G.S. (2012). Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications. Birkhäuser, Boston.

Chirikjian, G.S., Kyatkin, A.B. (2001). Engineering Applications of Noncommutative Harmonic Analysis: With Emphasis on Rotation and Motion Groups. CRC Press, Boca Raton.

Chua, C.L., Calandra, R., McAllister, R., Levine, S. (2018). Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. Advances in Neural Information Processing Systems (NeurIPS), 31.

Craig, J.J. (2005). Introduction to Robotics: Mechanics and Control, 3rd edition. Pearson Prentice Hall, Upper Saddle River, NJ.

Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function. Mathematics of Control, Signals, and Systems, 2(4), 303–314.

Featherstone, R. (2008). Rigid Body Dynamics Algorithms. Springer, New York.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., Davidson, J. (2018). Learning Latent Dynamics for Planning from Pixels. Proceedings of the 36th International Conference on Machine Learning (ICML), 2555–2565.

Hafner, D., Lillicrap, T., Norouzi, M., Ba, J. (2020). Dream to Control: Learning Behaviors by Latent Imagination. International Conference on Learning Representations (ICLR).

Hafner, D., Lillicrap, T., Ba, J., Norouzi, M. (2021). Mastering Atari with Discrete World Models. International Conference on Learning Representations (ICLR).

Hafner, D., Pasukonis, J., Ba, J., Lillicrap, T. (2023). Mastering Diverse Domains through World Models. arXiv:2301.04104.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. Proceedings of the 35th ICML, 1861–1870.

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of CVPR, 770–778.

Hornik, K., Stinchcombe, M., White, H. (1989). Multilayer Feedforward Networks Are Universal Approximators. Neural Networks, 2(5), 359–366.

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., Hutter, M. (2019). Learning Agile and Dynamic Motor Skills for Legged Robots. Science Robotics, 4(26), eaau5872.

Jankovic, M. (2018). Robust Control Barrier Functions for Nonlinear Systems with Input Delay. IEEE Transactions on Automatic Control, 63(4), 913–928.

Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G. (2023). 3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM Transactions on Graphics, 42(4).

Khalil, H.K. (2002). Nonlinear Systems, 3rd edition. Prentice Hall, Upper Saddle River, NJ.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R. (2017). Overcoming Catastrophic Forgetting in Neural Networks. Proceedings of the National Academy of Sciences, 114(13), 3521–3526.

LeCun, Y. (2022). A Path Towards Autonomous Machine Intelligence. Open Review Preprint. https://openreview.net/forum?id=BZ5a1r-kVsf.

Lee, T., Leok, M., McClamroch, N.H. (2010). Geometric Tracking Control of a Quadrotor UAV on SE(3). Proceedings of the 49th IEEE Conference on Decision and Control, 5420–5425.

Lee, T., Leok, M., McClamroch, N.H. (2018). Global Formulations of Lagrangian and Hamiltonian Dynamics on Manifolds. Springer, Cham.

Lipman, Y., Chen, R.T.Q., Ben-Hamu, H., Nickel, M., Le, M. (2022). Flow Matching for Generative Modeling. arXiv:2210.02747.

Long, A.W., Wolfe, K.C., Mashner, M.J., Chirikjian, G.S. (2012). The Banana Distribution is Gaussian: A Localization Study with Exponential Coordinates. Robotics: Science and Systems (RSS), Sydney, Australia.

Lutter, M., Silberbauer, J., Watson, J., Peters, J. (2020). Differentiable Physics Models for Real-world Offline Model-Based Reinforcement Learning. arXiv:2011.01734.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D. (2015). Human-level Control through Deep Reinforcement Learning. Nature, 518, 529–533.

Murray, R.M., Li, Z., Sastry, S.S. (1994). A Mathematical Introduction to Robotic Manipulation. CRC Press, Boca Raton. (Available online at: https://www.cds.caltech.edu/~murray/mlswiki.)

Pathak, D., Agrawal, P., Efros, A.A., Darrell, T. (2017). Curiosity-Driven Exploration by Self-Supervised Prediction. Proceedings of the 34th ICML, 2778–2787.

Rezende, D.J., Mohamed, S., Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. Proceedings of the 31st ICML, 1278–1286.

Romero, B., Fang, H.S., Agrawal, P., Bhatt, U. (2024). Eyesight Token: Generative Tactile-Visual Pretraining via Real and Simulated Correspondences. arXiv:2407.00439.

Ross, S., Gordon, G., Bagnell, J.A. (2011). A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS), 627–635.

Schulman, J., Levine, S., Moritz, P., Jordan, M.I., Abbeel, P. (2015). Trust Region Policy Optimization. Proceedings of the 32nd ICML, 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G. (2009). Robotics: Modelling, Planning and Control. Springer, London.

Sontag, E.D. (1983). A Lyapunov-Like Characterization of Asymptotic Controllability. SIAM Journal on Control and Optimization, 21(3), 462–471.

Spong, M.W., Hutchinson, S., Vidyasagar, M. (2006). Robot Modeling and Control. Wiley, Hoboken, NJ.

Sutton, R.S., McAllester, D., Singh, S., Mansour, Y. (1999). Policy Gradient Methods for Reinforcement Learning with Function Approximation. Advances in Neural Information Processing Systems (NeurIPS), 12.

Sutton, R.S., Barto, A.G. (2018). Reinforcement Learning: An Introduction, 2nd edition. MIT Press, Cambridge, MA.

Tassa, Y., Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T., Heess, N. (2018). DeepMind Control Suite. arXiv:1801.00690.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P. (2017). Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 23–30.

Todorov, E., Erez, T., Tassa, Y. (2012). MuJoCo: A Physics Engine for Model-Based Control. IEEE/RSJ IROS, 5026–5033.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems (NeurIPS), 30.

Williams, R.J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. Machine Learning, 8(3–4), 229–256.

Yoshikawa, T. (1985). Manipulability of Robotic Mechanisms. The International Journal of Robotics Research, 4(2), 3–9.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J.Y., Levine, S., Finn, C., Ma, T. (2020). MOPO: Model-based Offline Policy Optimization. Advances in Neural Information Processing Systems (NeurIPS), 33.

Zhu, Y., Wong, J., Mandlekar, A., Martín-Martín, R., Joshi, A., Nasiriany, S., Zhu, Y. (2020). robosuite: A Modular Simulation Framework and Benchmark for Robot Learning. arXiv:2009.12293.